Received December 12, 2020, accepted December 18, 2020, date of publication December 22, 2020, date of current version January 4, 2021.

Digital Object Identifier 10.1109/ACCESS.2020.3046693

Optimizing the Lifetime of Software Defined Wireless Sensor Network via Reinforcement Learning

MUHAMMAD USMAN YOUNUS[1], (Member, IEEE), MUHAMMAD KHURRAM KHAN[2], (Senior Member, IEEE), MUHAMMAD RIZWAN ANJUM[3], SHARJEEL AFRIDI[4], (Senior Member, IEEE), ZULFIQAR ALI ARAIN[5], AND ABDUL ALEEM JAMALI[6]

[1]Ecole Doctorale Mathéematiques, Informatique, Telecommunications de Toulouse, Paul Sabatier University, 31330 Toulouse, France
[2]Center of Excellence in Information Assurance, College of Computer and Information Sciences, King Saud University, Riyadh 11451, Saudi Arabia
[3]Department of Electronic Engineering, The Islamia University of Bahawalpur, Bahawalpur 63100, Pakistan
[4]Department of Electrical Engineering, Sukkur IBA University, Sukkur 65200, Pakistan
[5]Department of Telecommunication Engineering, Mehran University of Engineering and Technology, Jamshoro 76062, Pakistan
[6]Department of Electronic Engineering, Quaid-e-Awam University of Engineering, Science and Technology (QUEST), Nawabshah 67480, Pakistan

Corresponding author: Muhammad Usman Younus (usman1644@gmail.com)

The work of Muhammad Khurram Khan was supported by the Researchers Supporting Project of King Saud University, Riyadh, Saudi Arabia, under Grant RSP-2020/12.

ABSTRACT Reinforcement learning (RL) is an unsupervised learning technique used in many real-time applications. The essence of RL is a decision-making problem. In RL, the agent constantly interacts with the environment and selects the next action according to previous feedback in terms of reward. In this paper, RL trains Software-Defined Wireless Sensor Networks (SDWSNs) controller to optimize the routing paths. We combine RL and SDN, where RL is applied to the SDN controller to generate the routing tables. We also propose four different reward functions for optimization of the network performance. RL-based SDWSN improves network performance by 23% to 30% in terms of lifetime compared with RL-based routing techniques. RL-based SDWSN performs well because it can intelligently learn the routing path at the controller. In addition, it has a faster network convergence rate than RL-based WSN.

INDEX TERMS Reinforcement learning, wireless sensor network, SDWSN, RL-based WSN, energy optimization, routing.

I. INTRODUCTION

The significance of Wireless Sensor Network (WSN) is increasing day by day because it consists of several tiny sensors deployed in different fields. Sensor nodes sense the environment, process the data, and transmit it to the remote base station (BS). Since the last decades, WSN has attained the research community's attention for its advantages such as ease of deployment, flexibility, scalability, and low cost. It is widely used in many applications such as health care, traffic control, structural monitoring, home applications (e.g., smart home and smart building), and many more [1]–[4]. However, it also has been used for environment monitoring, disaster areas, and military applications. For example, sensor nodes are dropped through drones due to the harsh environment for monitoring forest temperature. On the battlefield, sensor nodes are deployed for the monitoring of military forces and its vehicle or to track their movement in hostile environments [5], [6]. These sensor nodes are deployed once time for a long period. The energy consumption of sensor nodes becomes a significant issue in WSN because of its small battery that cannot be recharge due to some dangerous environment applications. It has some resource limitations, such as energy management, communication capability, low memory, security, heterogeneity, complexity, etc. Hence, Routing is an essential means to improve the energy consumption of WSNs.

Routing is a process of selecting the path for sending data from the source (sensor node) to the destination (sink/BS). Some architectures such as Software Defined Networking

The associate editor coordinating the review of this manuscript and approving it for publication was Chun-Wei Tsai.

(SDN) can be helpful for optimizing the routing in WSNs [7]. SDN is an emerging architecture with flexibility, dynamics, and low management cost. It separates the data plane from the control plane. The data plane contains the network nodes, while the control plane includes the network controller. SDN controller can view the underlying network globally, which controls the whole network efficiently. Because of its flexible architecture, SDN has been widely used in many modern networking applications [2], [8]. However, it has some limitations (i.e., finding the best routing path in real-time applications); thereby, reducing the network performance.

Reinforcement Learning (RL) can be a promising learning technique to find the best routing path in real-time applications. It is a machine learning technique in which a learner known as an agent selects the actions through interaction with the environment based on the current state. The agent takes actions in such a way to maximize the long-term reward [9]. However, in the traditional routing protocols [10], [11], the node uses a pre-established routing path for data transmission. It does not reflect the exact status of the current network, in which routing tables are established in advance. In RL-based algorithm, a Q-value is assigned to each possible action that indicates the goodness of an action. The agent selects one action according to Q-value during the learning process. After each round, the agent gets the reward based on the previous action to update Q-value, as shown in Figure 1. Over time, the agent learns the network behavior, changes the routing path according to the network situation, and gets the optimal path after some iterations in real-time. Hence RL based algorithms give better performance than non-RL based routing (traditional) algorithms.

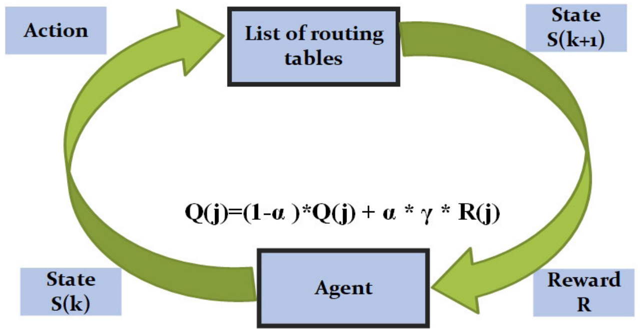

FIGURE 1. RL Model.

As we already put RL into practice in chapter 4 of [12]. However, there are some limitations in RL (i.e., convergence rate, excessive use of control packets for learning purposes, etc.), which also affect the network performance. However, the combination of SDN and RL can improve network performance.

Throughout this paper, we use RL to optimize the routing in SDWSN. SDN controller acts as an agent to learn from network behavior (environment) in real-time. SDN controller learns from a previously received response (reward) and takes the future actions. It learns how network energy is used for the selected actions. The RL-based SDWSN architecture is shown in Figure. 2. The main contributions of this paper are given below.

- We propose four reward functions for RL-based SDWSN to improve the network's lifetime and reduce the network's energy consumption.
- Different routing paths are established through spanning tree protocol (STP) that gives the looping free paths. Three different types of STPs are used to get the routing paths: all possible ST, distance-based minimum ST, and hop-based minimum ST.
- The SDN controller selects the best routing path by using RL.
- Develop a testbed for an experimental work and analyze the performance of RL-based SDWSN techniques for WSN routing purposes.
- The implementation of RL-based SDWSN for WSN routing is performed on a real-testbed.
- A comparative analysis of RL-based SDWSN experimental work with an RL-based WSN technique on a real-testbed.

The rest of this paper is structured as follows: In Section II, the state-of-the-art provides an overview of RL-based routing in SDWSN. Section III explains the proposed methodology including the reward functions and algorithms. In Section IV, the energy consumption model for communication is explained. Section V provides the detail of testbed used for experimental work. The graphical results also present the performance of the proposed reward functions and compare with RL-based WSN routing techniques results. Finally, the paper is concluded in Section VI.

II. RELATED WORK

SDN has been used in many applications (i.e., routing, security, multimedia, etc.) because of the decoupling of the control plane from the data plane. SDN controller can improve the network performance from a global perspective, including network lifetime, jitter, transmission delay, load balancing, video quality, and bandwidth. However, sometimes, the SDN controller cannot manage the network efficiently. Machine Learning (ML) techniques can be implemented on the SDN controller to manage the network resources efficiently. In literature, some authors use a combination of SDN and RL to improve network performance. Here's a summary of some of them. In [13], an energy-efficient SDWSN cognitive prototype was developed for environmental monitoring applications. The proposed prototype is based on RL to process the information on the control plane. It improves the self-adaptability and energy efficiency of SDWSN by an interaction between agents and environments that enhance intelligence in policymaking. It manages the complex data fusion centrally through the SDN controller, where the low-complexity computation is performed on a data plane. This prototype takes into account the constrains of WSNs, where the low energy consumption and high computational capability are required. An irregular cellular learning

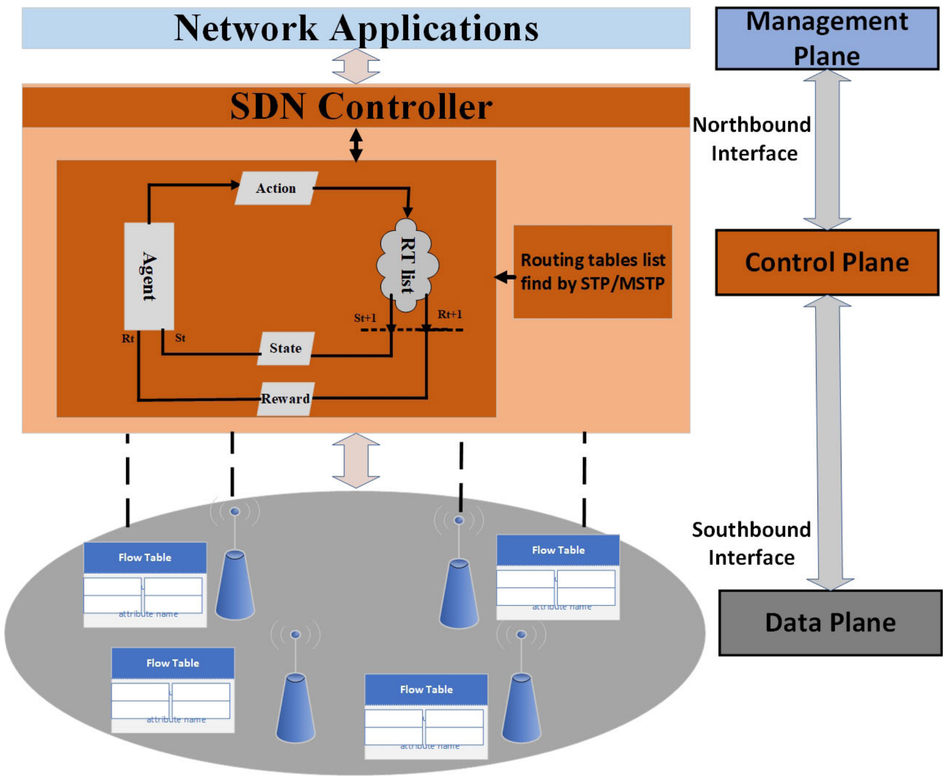

FIGURE 2. **RL-based SDWSN architecture.**

automaton (ICLA)-based algorithm in [14] has proposed, which examines self-protection issues with minimal energy consumption. It also schedules the sensors correctly into either an active or idle state. The network's sensing graph determines the minimum number of nodes through a Self-Protection Learning Automaton (SPLA) algorithm to secure the nodes. On average, it performs 50 % better than the maximum independent set and minimally connected dominant set algorithms despite having twice the energy consumption. There are still some issues that need to be addressed, While some solutions work well to reduce the delivery delay or expand the network. We are curious about such a routing protocol that takes into account both network lifetime and transmission delay. RL method is an effective method to solve this problem. The authors in [15] explored the less congested paths in the SDN network that were focussed on the Q-learning approach. Compared with Dijkstra and Extended Dijkstra, this method gets better results when the size of the transmitted data is increased.

Routing optimization is one of the significant traffic engineering control issues. Deep Reinforcement Learning (DQL) in [16] was used to reduce the transmission delay by optimizing the routing. The DQL agent characterizes the ways for all source-destination pairs at the network controller by communicating with the network environment. DRL network uses traffic matrix as the state represented by the bandwidth request between each source-destination pair, and the mean of end-to-end delays as the reward. It should be noted that the network factors (such as link quality, nodes' queue size, etc.) do not take into account because the scheme only considers the traffic matrix. The actor-critical approach is leveraged by DQL agents to solve the routing problem by automatically adjusting routing configurations to current traffic conditions. The routing results will be optimized by considering the other conditions. For testing the performance, OMNet++ discrete even simulator [17] is used. Simulation results show that DQL agents in a single phase can generate a near-optimal routing configuration. The proposed approach is attractive because the traditional optimization-based approaches involve many steps in generating a new configuration. A machine learning-based framework name as ''DROM'' for SDN is proposed in [18] to optimized routing, and improves network performance in terms of delay and throughput. The authors in [19] proposed an RL technique in Ubuntu Net Alliance's, the regional network for Southern and Eastern African National Research and Education Networks (NRENs), for SDN-based traffic engineering. This configuration needs to limit the use of different paths for contiguous data frames that may be addressed with large packet bucket sizes, and reduces the number of contiguous packets to solve a high jitter's key problem. The performance of the reward function is determined based on distance, the available link capacity, and the number of flows at the next hop. Q-Learning approach implementation involves two tables, such as a local Q-value table for each network node and a global aggregation table managed by the network controller. QoS measurements are performed by each switch at its next neighbors to transfer them to the controller. The controller uses active measurement data and interface-level statistics to update Q-values that depend on the calculation of reward values. The probabilistic selection of the forwarding link chooses the optimal path at each switch based on Q-values. Using Mininet emulator, the Q-learning approach distributes the traffic across links of multiple forwarding to maximize the throughput and to reduce the latency. Owing to the increase in Internet traffic, the latest developments in traffic engineering provide a range of strategies to solve network problems. Dynamic path planning is important in traffic engineering such as load balancing, traffic control, and a firewall. Therefore, a learning-based network path planning approach in [20] has presented for efficient traffic engineering under forwarding constraints. In the path planning problem, a sequence-to-sequence (Seq2Seq) model is used to learn implicit paths based on empirical network traffic data. Beam search is tailored that adapts the primary nodes sequence characteristics to improve the model efficiency, and provides the path connectivity. Mininet emulator environment was used to validate the derived model's effectiveness. It also trains and evaluates the model by leveraging the traffic data collected by both a real-world GEANT and a grid network topology. Experiment results show a high testing accuracy.

The next-generation network paradigm has immense advantages over traditional networks because it simplifies the management layer, especially by adopting SDN. Still, enabling QoS provision poses challenges, particularly for multimedia based applications. LearnQoS, introduced by [21], is an RL-based system that uses Q-learning to optimize QoS specifications for policy-based network management (PBNM) in multimedia-based SDNs. The RL model is defined using three elements, such as state, action, and reward. The traffic matrix represents the state, and the agent is taken into account by four different actions: doing nothing, reducing the data rate, increasing the data rate, and rerouting. Rewards, by contrast, are based on service level agreements (SLAs). Experimental results show that despite the network overhead introduced by LearnQoS, QoS performance is significantly improved compared to the default multimedia based SDN. In [22],the authors proposed a distributed SDN-based RL routing. Though the routing protocol was implemented in an SDN topology and operated in a distributed manner. RL algorithm was tested by adding it to the OSPF routing protocol. They compared it with the traditional OSPF routing protocol and achieved better QoS delays and jitter. An approach to routing based on the RL algorithm has been proposed in [23]. The optimization was implemented in a multi-layer hierarchical SDN architecture. The distributed hierarchical control plane architecture reduces the signaling delay in SDN by designing three different levels of controllers; the super, domain/master, and slave controllers.

Each controller has its specific responsibility group that minimizes the load of super controllers. The slave controllers provided read-only access to the data plane and received port messages from them. Also, the slave controllers carried out basic control functions, including traffic entry, flow control, and so on. The flow set-up requests were received by domain controllers to install the flow rules on switches. To achieve the optimal routing solutions, a QAR algorithm with QoS-based was implemented in the super controller with a global network's view and controlled full network functions. The propose algorithm had been implemented and tested in a real Sprint GIP network. Compared with existing learning solutions, the proposed approaches gave quick convergence. Also, in [18], a Deep Reinforcement Learning (DRL) approach called DDPG is introduced to identify the optimum routing solutions. The optimization process described in the paper is DROM, and experimental results show that the algorithm has high convergence and efficiency. In SDN, link quality requirements for service are not considered. A routing algorithm is proposed by the combination of RL and neural networks for SDN [24]. In RL, the agents continuously explore the surrounding environment without any prior environment. After a certain training cycle, the agent becomes able to choose the right action, but it needs manual design to train the Q-matrix for the convergence to corresponding features. Due to defects of artificial design in Q-learning, the Q-matrix is replaced by neural networks with the processing power for massive data. The proposed algorithm aims to select the best link among many optional links to improve the network QoS. After each round, the algorithm gets the reward against each selected link. The link performance is divided into three levels (when link QoS performance is 0-30%, 31-60%, and 61-100%, reward R is 50,100 and 150, respectively). The link performance level is used in reward function and selects the next action according to previously selected link performance. The SDN network performance can be improved by resolving the SDN controller's scalability issue on the control plane. In [25], the author proposed an RL-based algorithm called Q-placement that speeds up the network convergence rate and guaranteed performance. It reduces the average accumulated service cost for the end-user by investigating the service placement problem on SDN switches. The performance of a Q-Learning algorithm equipped with the SDN controller has investigated in [26] to determine the adapt video quality and to re-route traffic for layered adaptive streaming.

Some researchers use traffic shaping approaches to restrict each client's allocated bandwidth and to force them to request a minimum bit rate. Markov Decision Method (MDP) has been used for system modeling. The experimental results have shown that the MDP based proposed model performance is better than the greedy-based and shortest routing path. The researchers in [27] have proposed MIND to learn the probability distribution of selecting the top-k best path through Reproducing Kernel Hilbert Space (REPS-RKHS) that is more suitable for real-world issues such as routing. MIND predicts the spatial-temporal traffic information or network conditions, where the policy generation module is designed by optimal routing policies that learn from data based on RL. In distributed SDN, the problem of controller synchronization as a Markov Decision Process (MDP) was investigated in [28] with a limited synchronization budget to determine the rules that support the benefits of controller synchronization over time. An RL-based algorithm uses the deep neural network (DNN) to represent its value function, called the Deep-Q (DQ) Scheduler, which provides nearly twofold performance improvement compared to the state-of-the-art SDN controller synchronization solutions. However, some authors [29] use the RL for autonomous cyber defense in SDN and also use RL to resolve the synchronization issues of multiple controllers [30], [31]. Several AI techniques used in the SDN context have been introduced in [32], including different security and placement issues. The research focuses on three main AI sub-fields: ML, Meta Heuristics, and Fuzzy inference systems. Therefore, we explored their various implementation areas and future use, and the changes made in the SDN paradigm using AI-based technologies. Significant efforts were made in [33] on Artificial Intelligence (AI) to enhance the routing and security capabilities in SDN. AI and ML give rise to systems performance because they are capable of operating themselves. Investigating the AI algorithms in SDN may lead to better network management, security, or routing in SDWSN, and promote more reliable networks. The benefits of using these AI in SDWSN have been considered to solve the challenges faced by WSN and improve their performance.

III. METHODOLOGY

In this section, the proposed methodology is described in detail. RL and SDN are combined to find the best routing path in real-time applications. The combination of SDN and RL provides an optimized routing path.

In RL-based SDWSN, the data plane is separated from the control plane. The control plane contains the controller that generates the control traffic and collects the data packets from data plane devices. The control traffic consists of the routing path followed by the nodes (SDN data plane devices) to send the data packets to the controller/sink.

Initially, the SDN controller finds all the possible combinations of routing paths (also known as routing tables) through the Spanning Tree Protocol (STP) that prevents the network cycling problems. After finding all possible routing paths, the controller selects one RT from all possible paths and broadcasts it to the nodes, obtaining nodes' status data after each round. To choose the best path, Q-learning is introduced into the SDN controller that learns the path in real-time. In the learning process, the RL agent selects an action from the set of actions and receives a reward after interacting with the environment. Through a learning process, the selection of optimal action policy is developed.

In the proposed methodology, the SDN controller act as an agent, selecting as an optimized RT (action) from a set of routing paths and sending it to the underlying network. After each learning round, the controller receives the reward in terms of loss estimated path lifetime (EPLT), and since the loss cannot be positive, the maximum reward can be zero. When the reward is received, the controller observes how much energy was consumed in the last round of selected RT. The detail of the proposed reward functions is given in subsection 1 to 4.

A. REWARD

A reward in RL is part of the environmental feedback. When an agent interacts with the environment, he can observe the state changes and reward signals through his actions if there is a change. The agent uses this reward signal (which can be positive for good actions and negative for bad actions) to conclude how to behave in a state. The design of the reward function is a very challenging task because it highly impacts on network performance.

In this paper, four reward functions are proposed for the optimization of network lifetime. In the proposed reward function, the first reward function finds to global reward; the second reward function gives the average reward. While the third and fourth reward functions provide the global weighted reward and average global weighted reward, respectively. The detail of each reward function is given below:

1) REWARD FUNCTION 1 (RF-1)

The global reward function is the concept of the estimated lifetime of the sum of all paths of nodes. The objective of the learner (agent) is to choose an action (selects one routing path from the routing list) after each round that minimizes the global path loss. Let there is N number of nodes, where each node estimate the remaining lifetime and send it to the controller after each round through selected forwarder. The intelligent controller receives a total reward 'R' after taking action is given as:

$$Loss^{EPLT}_{Globla} = \sum_{i=0}^{N}(EPLT_i - ENLT_i) \tag{1}$$

N is the total number of nodes in the network, where the *EPLT* and *ENLT* are estimated path lifetime and estimated node lifetime, respectively.

2) REWARD FUNCTION 2 (RF-2)

It gives the average loss of the network and provides better results as compared to RF-1 because of the average of all received loss from all the network nodes.

$$Loss^{EPLT}_{Average} = \frac{1}{N}\sum_{i=0}^{N}(EPLT_i - ENLT_i) \tag{2}$$

3) REWARD FUNCTION 3 (RF-3)

Any weighted parameter can improve the network performance. In the third reward function, we include the *Distance_to_Sink* parameter to see the impact on *EPLT* loss. Here the combined reward is taken into account with the weighted parameter (*Distance_to_Sink*) of each node. The mathematical expression of global reward in the form of

EPLT loss is as follows:

$$Loss^{EPLT}_{GlobalWeighbt} = \sum_{i=0}^{N}(EPLT_i - ENLT_i) * \frac{1}{Dis.ToSink_i} \quad (3)$$

Dis._to_Sink is the distance from node *i* to sink, and *N* is the total number of nodes in the network. *EPLT* and *ENLT* are estimated path lifetime and estimated node lifetime, respectively.

4) REWARD FUNCTION 4 (RF-4)

It gives the average weight loss of the network and provides the better result as compared to RF-3 because of the average of all received loss from all the network nodes.

$$Loss^{EPLT}_{AverageWeight} = \frac{1}{N}\sum_{i=0}^{N}(EPLT_i - ENLT_i) * \frac{1}{Dis.ToSink_i} \quad (4)$$

B. Q-VALUE

The initial Q-value is considered as the worst case where all the nodes die without sending any data, and it changes according to the reward function. *R* is the reward that comes from four different reward functions based on the selection reward function, as described in section 3.1. The reward is in the form of an *EPLT* loss; that is why, the maximum reward will be zero because the loss cannot be positive.

$$Q(j) = (1 - \alpha) * Q(j) + \alpha * \gamma * R(j) \quad (5)$$

α, γ, and *R* is the learning rate, discount factor and reward, respectively.

Initially, Q-value of RF-1, RF-2,RF-3, and RF-4 can be get from Eq. 6, 7 8, 9.

$$Q(j) = -\sum_{i=0}^{N}(ENLT_i) \quad (6)$$

$$Q(j) = -\frac{1}{N}\sum_{i=0}^{N}(ENLT_i) \quad (7)$$

$$Q(j) = -\sum_{i=0}^{N}(ENLT_i) * \frac{1}{Dis.ToSink_i} \quad (8)$$

$$Q(j) = -\frac{1}{N}\sum_{i=0}^{N}(ENLT_i) * \frac{1}{Dis.ToSink_i} \quad (9)$$

The network controller learns from each round and follows a best spanning tree path, giving the minimum loss and optimal Q-value. The minimum *EPLT* loss means that the network follows the best routing path.

C. ALGORITHM DETAIL

The RL-based SDWSN experiments follow the procedure given in section V-A. The experimental work is based on two different planes: control plane and data plane. The control plane includes the intelligent SDN controller that also used as a sink to collect the data from data plane, as shown in Figure 3.

First, the SDN controller generates the routing table (RT) through the STP. It selects an RT from the RT list to broadcast it. After each round, the controller also collects the node status data. SDN controller calculates the reward by selected reward function (RF1 to RF4), and computes the Q-Value after each learning round and selects the next RT according to received reward. For the initially Q-value, we have considered the worst-case scenario when all nodes die without sending data and use the global *EPLT* loss as Q-Value. After the initial round, the SDN controller uses the status data (node data) to calculate the Q-value.

After each round, if the controller observes that any node runs out of battery, it excludes the node from RT and then recalculates the routing paths through the STP /Minimum Spanning Tree (MST) function. It follows the same procedure until the last node dies. On the other side, the data plane contains the sensor nodes that receive the tunable parameters and RT from the SDN controller. It follows the same path for sending the data packet up to the destination (controller). The node also receives the data packet from a neighboring node. Then the receiving node checks the receiving address of the packet, and if the current/receiver node is the forwarder of a received data packet, it checks the destination address and puts the data packet into a transmission queue; otherwise, it will discard it.

D. ALGORITHM

1) ALGORITHM FOR SDN CONTROLLER

1) Assign the IP and broadcast the address to an SDN controller after its initialization.
2) Find all the routing paths by STP (all possible STs [34]), Distance-based MST (using **MST Function**), Hop-based MST (using **MST Function**) according to selected spanning tree protocol and then put them into the routing path list.
3) Initially the Path Estimate Lifetime (PELT) would be zero.
4) SDN controller selects one routing RT from the routing path list and broadcasts it.
5) SDN controller receives the status data that includes the NELT of each node at the end of the round. It observes the *EPLT* and sees how much energy has been consumed in the current round for the selected RT.
6) Estimate the path lifetime.
7) Calculate the reward by selected reward functions (RF1 to RF4), compute the Q-Value after each learning round, and select the next RT according to received reward. However, for the initially Q-value, we considered the worst-scenario where all nodes die without sending data, and used the global EPLT loss as Q-Value.
8) At the end of each round, the SDN controller excludes the node from RT, which dies in the last round, and then STP recalculates the routing paths according to Step 2.
9) Repeat Step 4 to 8 until the last node dies.

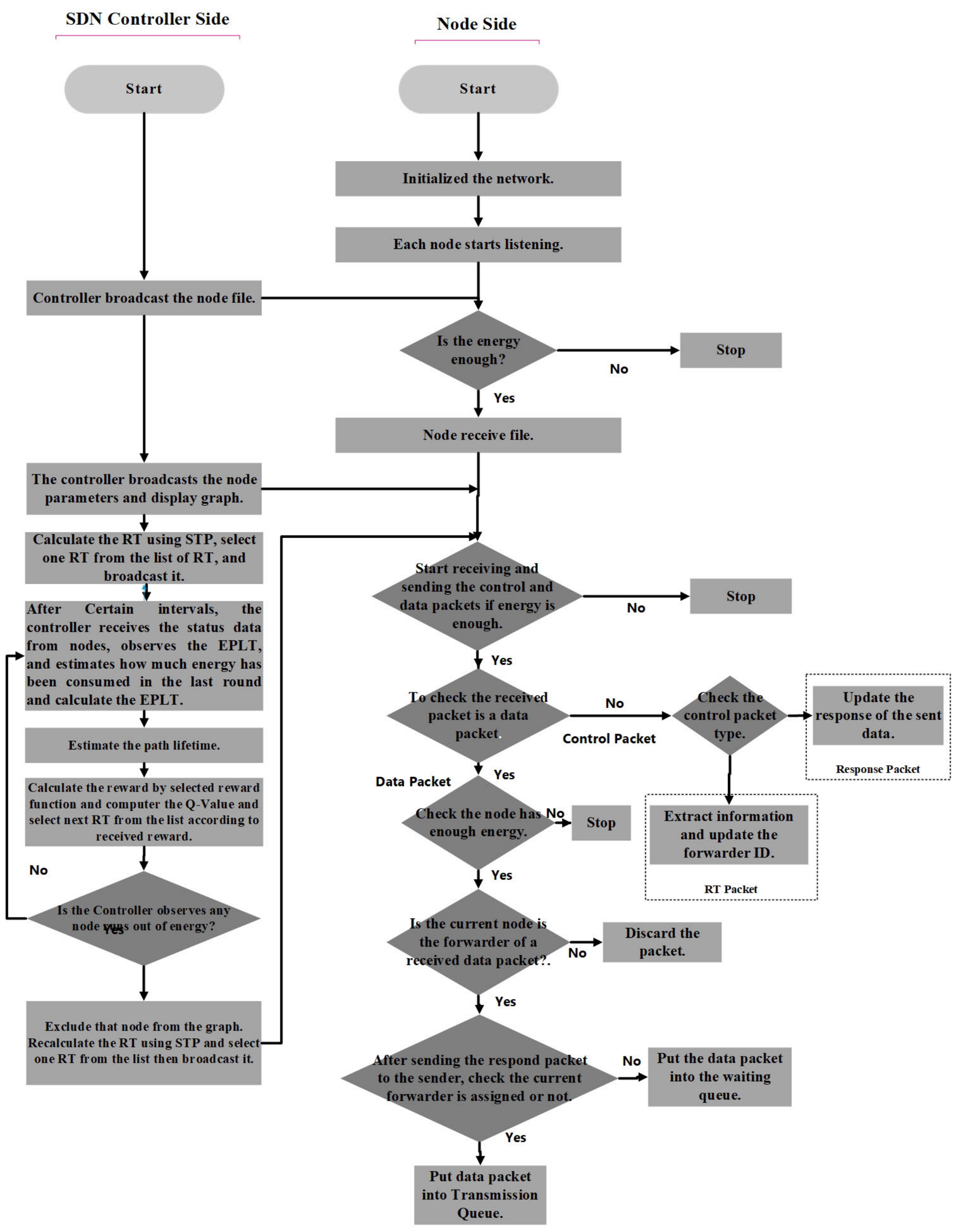
SDN Controller Side
Node Side
Start
Start
Initialized the network.
Each node starts listening.
Controller broadcast the node file.
Is the energy enough?
No
Stop
Yes
Node receive file.
The controller broadcasts the node parameters and display graph.
Calculate the RT using STP, select one RT from the list of RT, and broadcast it.
Start receiving and sending the control and data packets if energy is enough.
No
Stop
Yes
After Certain intervals, the controller receives the status data from nodes, observes the EPLT, and estimates how much energy has been consumed in the last round and calculate the EPLT.
To check the received packet is a data packet.
No
Control Packet
Check the control packet type.
Update the response of the sent data.
Response Packet
Estimate the path lifetime.
Data Packet
Yes
Calculate the reward by selected reward function and computer the Q-Value and select next RT from the list according to received reward.
Check the node has enough energy.
No
Stop
Extract information and update the forwarder ID.
RT Packet
No
Yes
Is the Controller observes any node runs out of energy?
Yes
Is the current node is the forwarder of a received data packet?.
No
Discard the packet.
Yes
Exclude that node from the graph. Recalculate the RT using STP and select one RT from the list then broadcast it.
After sending the respond packet to the sender, check the current forwarder is assigned or not.
No
Put the data packet into the waiting queue.
Yes
Put data packet into Transmission Queue.

FIGURE 3. Flow chart of RL based SDWSN working.

2) ALGORITHM FOR NODE

1) Assign the IP and broadcast address to a node after its initialization.
2) Get the desired parameters set by an SDN controller.
3) Assign the received parameters setting to the corresponding parameter.
4) Receive the routing path from the SDN controller.
5) Calculate the initial value of the average Control Receiving (CT-Rx) load.
6) Estimate the initial value of Avg DTx-Rx load.
7) Calculate the Average Transmitted and Received (Avg DTx-Rx) data of each round.
8) Estimate the residual battery capacity (BC).
9) Estimate the node lifetime.
10) Establish the Rx and Tx queue of control and data packets.
11) Check the neighbors status. Also, call the CBR Data Packet function to transmit the data packet of a node.
12) Check the node battery is sufficient or not.

13) If the battery is sufficient, the received control (Ctl) and data (DT) packets are queued according to the packet type. Handle the received data (Forward/Drop) and control packets (Store) based on the received packet status.

3) MINIMUM SPANNING TREE (MST) FUNCTION

1) Sort all the edges in the non-decreasing order of their weight.
2) Pick the smallest edge.
3) Check either it forms a cycle or not using **Cycle Detection Function**.
4) If the cycle is not formed then include this edge into MST; otherwise, discard it.
5) Repeat Step 2 to 4 until there are (Vertices-1) edges in the spanning tree.

4) CYCLE DETECTION FUNCTION

1) For each edge, make subsets using both vertices of the edge.
2) If both vertices are in the same subset then a cycle is found; otherwise, no cycle has found.

IV. ENERGY MODEL

In wireless communication, each sensor node consumes a certain amount of energy to transmit and receive the packets. RL-based SDWSN experimental work uses the energy consumption model to calculate the energy consumption of each node after each leaning period and also estimates the remaining node lifetime [12]. Each node's energy consumption depends on the total transmitted and received data during each round.

In each learning period, the node consumption is calculated based on total transmitted and received data. The node consumption and remaining lifetime are calculated as following:

$$t_{Tx}^{Ctl} = D_{AvgTx}^{Ctl} * 8/R_b \tag{10}$$

t_{Tx}^{Ctl} is the control bits transmission time,where the D_{AvgTx}^{Ctl} and R_b represent the control average transmitted data and bit rate (bps) of node, respectively.

$$I_{Tx}^{Ctl} = t_{Tx}^{Ctl} * (P_{Tx}^{max}/V_{out}/1000) \tag{11}$$

I_{Tx}^{Ctl} represents the current used for control bits transmission during the learning period, where the P_{Tx}^{max} and V_{out} indicate the maximum power used for transmission and transceiver output voltage, respectively. Here the power is taken in mW.

$$t_{Tx}^{data} = D_{AvgTx}^{data} * 8/R_b \tag{12}$$

t_{Tx}^{data} denotes the data bits transmission time, where the D_{AvgTx}^{data} and R_b represent the average transmitted data and bit rate (bps) of node, respectively.

$$I_{tx}^{data} = t_{Tx}^{data} * (P_{Tx}/V_{out})/1000 \tag{13}$$

I_{Tx}^{data} represents the current used for data transmission during the learning period, where the P_{Tx} is the power used for transmission that depends on selected neighbor (current forwarder) distance, and V_{out} denotes the transceiver output voltage.

$$t_{Rx}^{Ctl} = D_{AvgRx}^{Ctl} * 8/R_b \tag{14}$$

$$t_{Rx}^{data} = D_{AvgRx}^{data} * 8/R_b \tag{15}$$

$$t_{Rx} = t_{Rx}^{Ctl} + t_{Rx}^{data} \tag{16}$$

t_{Rx}^{Ctl} ,D_{AvgRx}^{Ctl}, D_{AvgRx}^{data}, and t_{Rx}^{data} are the control bits receiving time, the average received control data, the average received data, and data bits receiving time, respectively. The t_{Rx} is the total receiving time of control and data traffic during the learning period.

$$I_{Rx} = t_{Rx} * (P_{Rx}/V_{in})/1000 \tag{17}$$

I_{Rx} denotes the current used for receiving control and data traffic during the learning period, where P_{Rx} represents the power used for receiving control and data traffic during the learning period, and V_{in} indicates the transceiver input voltage.

$$t_{idle} = t_{Lp} - t_{Tx}^{Ctl} - t_{Tx}^{data} - t_{Rx} \tag{18}$$

t_{idle} is an idle time during the learning period, which is getting from subtracting the control and data traffic transmission time period, including total reception time from the learning period. While t_{LP} indicates the learning period.

$$I_{idle} = t_{idle} * (P_{idle}/V_{idle})/1000 \tag{19}$$

P_{idle} and V_{idle} are the power and voltage used during idle time, respectively where the I_{idle} is the current used during idle time period.

$$I = I_{Tx}^{Ctl} + I_{Tx}^{data} + I_{Rx} + I_{idle} \tag{20}$$

I is the total used current during transmission, reception, and idle periods.

$$I_{proc} = t_{Lp} * (P_{proc}/V_{system})/1000 \tag{21}$$

I_{proc} and P_{proc} indicate the current and power used during processing, respectively. Assume that the processing energy consumption per time unit is constant.

$$ERBC = (1 - SDF) * ERBC - I - I_{proc} \tag{22}$$

$ERBC$ and SDF is the estimate residual battery capacity and self-discharge factor, respectively.

$$NELT = ERBC * t_{Lp}/I \tag{23}$$

$NELT$ denotes the node estimate lifetime. It is calculated through estimate residual battery capacity, learning period, and total used current

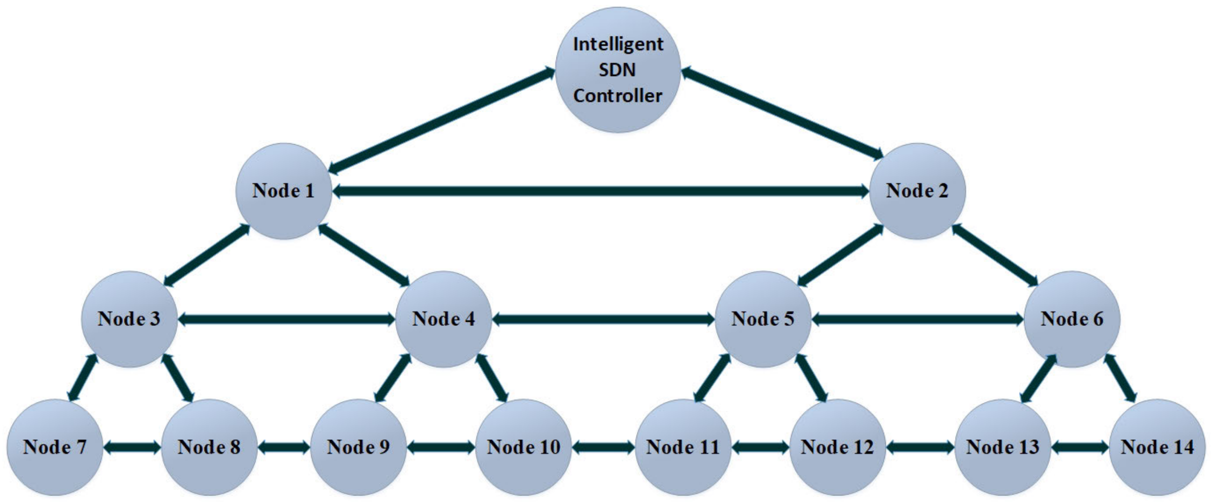

FIGURE 4. Graph used for RL-based SDWSN experimental work.

V. EXPERIMENTAL RESULTS AND DISCUSSION

A. EXPERIMENTAL SETUP AND PLATFORM

We have performed our experiments on the real-testbed using Python 3.0. Raspberry Pi 3 has used that is low power and low-cost device with small size single-board computer. In the experiments, the ad-hoc network is developed by using 802.11ac wireless LAN. Each node is connected to its neighboring node through the wireless LAN. In our experimental scenario, we used fourteen nodes, all in the same location, and logical distances are used to place the graph instead of real deployment. The graph used for experimental work is shown in Figure 4. During experimental work, the energy consumption is calculated through simulation because Raspberry Pi could not directly measure the remaining battery capacity of nodes. The energy consumption model has explained in section 5. The simulation parameters are the same as those used for the experimental work based on RL, as given in section 6.2.

B. EVALUATION METRICS

In this paper, we consider two metrics for results evaluation, as describe below:

1) NETWORK LIFETIME (LT)

It is a time duration until the last node (the sink is not reachable) dies.

2) CONVERGENT RATE

The rate at which the convergent sequence of a network approaches to its limit.

C. RESULTS AND DISCUSSION

We conducted experiments to evaluate the performance of RL-based SDWSN and compare it with RL-based routing techniques. In the graphical representation, we use some general labels for the depiction of each combination curve instead of a full technique name because RL-based routing techniques names are too long. We try to write some techniques names in the graphical representation as well, which is possible. Each combination detail is given below:

1-1 $\rightarrow$ All accept - Epsilon greedy selection.
1-2 $\rightarrow$ All accept - Adaptive e-greedy selection.
1-3 $\rightarrow$ All accept - e-multicriteria selection.
1-4 $\rightarrow$ All accept - Adaptive e-multicriteria selection.
2-1 $\rightarrow$ Distance based accepts - Epsilon greedy selection.
2-2 $\rightarrow$ Distance based accepts - Adaptive e-greedy selection.
2-3 $\rightarrow$ Distance based accepts - e-multicriteria selection.
2-4 $\rightarrow$ Distance based accepts - Adaptive e-multicriteria selection.
3-1 $\rightarrow$ Quality-conserving-aware accept - Epsilon greedy selection.
3-2 $\rightarrow$ Quality-conserving-aware accept - Adaptive e-greedy selection.
3-3 $\rightarrow$ Quality-conserving-aware accept - e-multicriteria selection.
3-4 $\rightarrow$ Quality-conserving-aware accept - Adaptive e-multicriteria selection.
4-1 $\rightarrow$ Load balancing-aware accept - Epsilon greedy selection.
4-2 $\rightarrow$ Load balancing-aware accept - Adaptive e-greedy selection.
4-3 $\rightarrow$ Load balancing-aware accept - e-multicriteria selection.
4-4 $\rightarrow$ Load balancing-aware accept - Adaptive e-multicriteria selection.

Each combination is composed of a request handling method (that handles the control packets) and neighbor selection method, which selects the next forwarder for sending data packets. In request handling methods, three different methods were proposed [12] that include "Distance-Based Accept", "Quality-Conserving-Aware Accept," and "Load-Balance-Aware accept". One traditional method,

TABLE 1. Experimental constant parameters.

Parameter Symbol	Value	Unit	Description
α	0.3		Alpha
Γ	1		Gamma
SDF	0.025		3% is the average self-discharge factor per month of Lithium batteries
BC	100	AS	The initial battery capacity of each node
$ProcPw$	3	mW	Processing power level of node
System voltage	5	V	Node processing voltage
$RecPw$	20	mW	Receiving power level of node
V_{in}	3	V	Transceiver input voltage
P_{idle}	20	mW	Power used during idle time period
V_{idle}	3	V	Transceiver idle voltage
V_{out}	5	V	Transceiver output voltage

TABLE 2. Experimental variable parameters.

Parameter Symbol	Value	Unit	Description
CBR_Packet_size	100,1000	bytes	CBR data traffic size
CBR_Period	1,5,10,20	s	CBR traffic period

"All accept" is also presented. However, in the neighbor selection side, three different methods are also proposed that include "Adaptive epsilon greedy selection", "multicriteria epsilon greedy selection," and "Adaptive multicriteria epsilon greedy selection". One traditional method, "Epsilon greedy selection" is also explained.

The first request handling methods is based on distance. It limits the number of forwarders to reduce energy consumption. Each node tries to build a path through that node, which is closest to a sink, and mathematical detail is given in [12]. In the quality conserving aware handling request, the degradation of QoS is considered before accepting the neighboring node request. The acceptance probability of new requesting nodes depends on the effect on their acceptance on the path estimation lifetime (PELT) of already accepted neighbors. The mathematical work is explained in [12]. However, the last request handling method is "Load balance aware accept". In this technique, each node accepts the neighboring request according to its load, leading to an increase in the network lifetime and PDR. A node with fewer followers and higher nodes estimate lifetime (NELT) compared to their neighbors is more likely to provide higher PELT to the requesting node and should be a high probability of accepting the neighboring request. The complete detail is given in [12].

However in neighbor selection methods, the adaptive epsilon greedy used dynamic epsilon and optimal convergence factor. Optimal convergence factor (Q_{cf}) calculates the ratio between the highest Q-value to the highest node lifetime. Q_{cf} goes to increase with the passage of time and when it is equal to 1, then an optimal path is found. When the Q_{cf} factor is close to 1, then the search may stay in local minimum. A random search is used to discover a better route which is close to optimal (that depends on required accuracy) route. If $1 - Q_{cf}$ is less than the required accuracy then the near-to-optimal route is found that requires the future changes. The mathematical detail is given in [12]. The second method for neighboring node selection was Multicriteria epsilon greedy selection. This method operates with three different scenarios.

- A random selection among an epsilon $\in$ fraction of all candidates.
- Greedy selection among a $1 - \frac{\in}{2}$ fraction of candidates with the product (QP) of highest Q-value ($Q_{highest}$), and Path Estimate LifeTime (PELT).
- Greedy selection among a $1 - \frac{\in}{2}$ fraction of candidate with the highest composed matric value of three parameters (i.e., Node estimate lifetime $NELT$, distance to sink (D_To_Sink), and number of hops (H_To_Sink)).

Three parameters $NELT$, D_To_Sink, and H_To_Sink, are used for the selection of forwarder, which is called weight-based selection. In weight-based selection, W_NELT is the weight of a lifetime of the neighbor node. The higher W_NELT can support to forward the additional data traffic. W_Dist is the weight of the euclidian distance of neighbor to the sink. The lower W_Dist neighbor node can save energy because of low distance and has more chances to choose that path. W_Hops is the weight of hop counts to sink. The lower W_Hops can also save the energy and likely to choose that path. A detailed explanation is given in [12]. However, the last neighbor selection method is adaptive multicriteria epsilon greedy selection. In this technique, dynamic epsilon is used for neighbor selection instead of the fixed epsilon that the RL agent use for exploration. The whole detail is given in [12].

In our paper, the experimental results are analyzed using different metrics and compared with RL-based techniques for WSN (all techniques are explained above), as described in the following subsections.

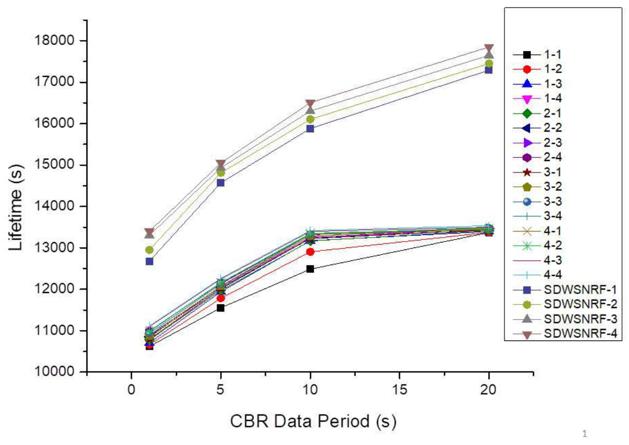

FIGURE 5. Comparison of RL-based SDWSN and RL-based WSN routing.

1) COMPARISON OF RL-BASED ROUTING FOR SDWSN AND RL-BASED DIFFERENT TECHNIQUES FOR WSN

Firstly, we compared the RL-based WSN routing performance with RL-based SDWSN routing, as shown in Figure 5. From graphical representation, it can be seen that the performance of RL-based SDWSN routing is better than RL-based WSN routing. In RL-based WSN routing, each node learns the routing path through the information of neighboring nodes. Each node shares the control information with its neighboring nodes and selects a forwarding node to send the data packets to the controller/sink. It takes a long time to learn the path. However, in RL-based SDWSN routing, the controller has information of the whole network. In the beginning, the controller generates a list of all possible routing paths (routing tables) through STP, and the controller selects a routing table from the list to get a reward in the form of *PELT* loss. Four reward functions are used for RL-based SDWSN experiments, namely RF-1, RF-2, RF-3, and RF-4. In Figure 5,the graphical representation shows that RF-4 based SDWSN experiments have the longest lifetime because it calculates the average weighted *EPLT* loss. The weighted parameter used in the reward function is the distance of each node to sink, which helps to make the reward function more efficient as compared to RF-1 and RF-2. However, the accumulative loss is taken in RF-3 and RF-4 is the average loss that gives higher Q-value.

2) COMPARISON OF RL-BASED ROUTING FOR SDWSN WITH DISTANCE-BASED MSTs AND DIFFERENT TECHNIQUES OF RL

In this scenario, the Distance-based Minimum Spanning Trees (MSTs) are used as routing tables in RL-based SDWSN routing. The graphical representation shows that the RL-based SDWSN routing with Distance-Based MSTs outperforms. Here the distance-based MST gives the low-cost routing path in terms of the shortest distance. MST provides few routing paths. It means that after a few epochs, the controller learns the optimal path in a short time and uses the shortest distance for sending the data packets, consuming less energy. As a result, it improves the network

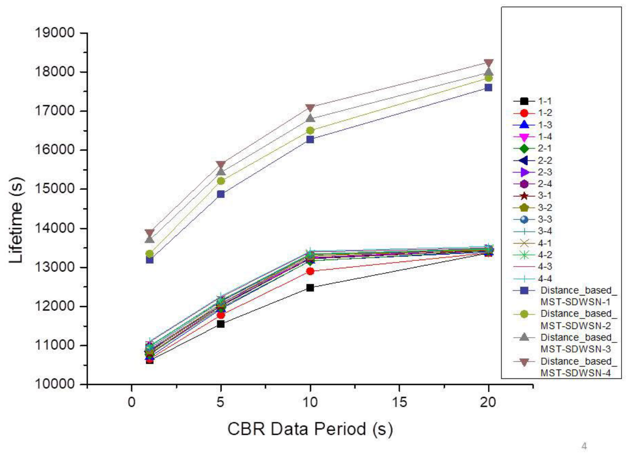

FIGURE 6. Comparison of distance-based MST-SDWSN and RL-based WSN network routing.

performance in terms of lifetime. It has shown from Figure 6 that the RL-based SDWSN experiment (named as distance-based MST-SDWSNRF-1, 2, 3, and 4) gives better performance as compared to RL-based WSN techniques because of shortest distance, low control traffic, and quick routing path established by centralized control.

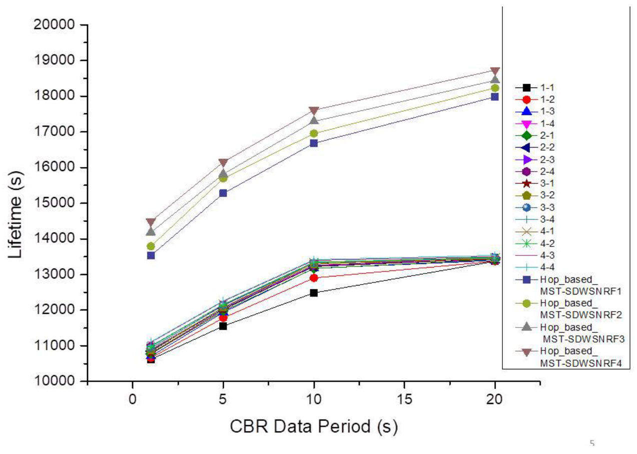

FIGURE 7. Comparison of hop-based MST-SDWSN and RL-based WSN network routing.

3) COMPARISON OF RL-BASED ROUTING FOR SDWSN WITH HOP-BASED MSTs AND DIFFERENT TECHNIQUES OF RL

In this section, the controller finds the routing paths through hop-based MST and compares the performance of RL-based SDWSN with RL-based WSN techniques as shown in Figure 7. The experimental results shows that RL-based SDWSN outperforms. In RL-based SDWSN, the routing path is found through hop-based MST. It gives the routing paths list that has a minimum number of hops from each node to the sink/controller.

Form the list of routing paths, the RL agent selects the best routing table based on the received reward that is in the form of *EPLT* loss. Hop-based MST gives a better routing path because each node follows a path with a minimum number of hops. In results, the processing energy consumption of relaying nodes has reduced that leads to enhance the network performance in terms of lifetime. The graphical results show that RL-based SDWSN experiments (named as SDWSNRF-1, 2, 3, and 4) give better performance as compared to RL-based WSN techniques due to less control traffic exchange and quick routing path established by SDN controller management and balanced network.

4) COMPARISON OF DISTANCE-BASED AND HOP BASED MSTs FOR SDWSN

For generating MSTs, two factors are used: the first is a distance, and the second is the number of hops. SDN controller selects an MST (that used as a routing table) from the MSTs list and broadcasts it to all nodes. It has been observed from the graphical representation that the performance of hop-based MSTs SDWSN gives better performance as compared to distance-based MSTs SDWSN in terms of the network, as shown in Figure 8 because the hop-based MST balances the network nodes energy more efficiently. Each node (hop) also reduces the processing energy consumption due to the minimum number of hops from source to destination. Besides, the load on the relaying node becomes a balance that leads to enhanced network lifetime. Hence hop-based MSTs SDWSN improves the network lifetime by 3% compared to distance-based MSTs SDWSN.

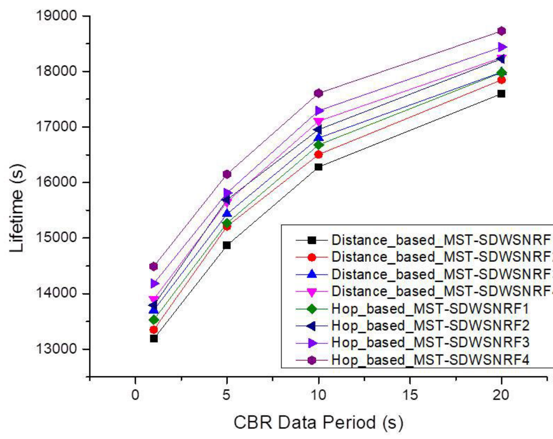

FIGURE 8. Comparison of distance-based MST-SDWSN and hop based MST-SDWSN network routing.

5) COMPARISON OF RL-BASED SDWSN AND DIFFERENT RL TECHNIQUES PERFORMANCE WITH DIFFERENT DATA RATES

In this section, we consider the data rate as a comparison parameter to observe the network lifetime. In the RL-based SDWSN experiment, data rates of 100 and 1000 bytes are used to analyze the data effect on network lifetime and compared with RL-based different techniques using the data rate of 100 bytes. From the graphical representation, we can see that the network performance of RL-based SDWSN is higher than that of RL-based different techniques at data

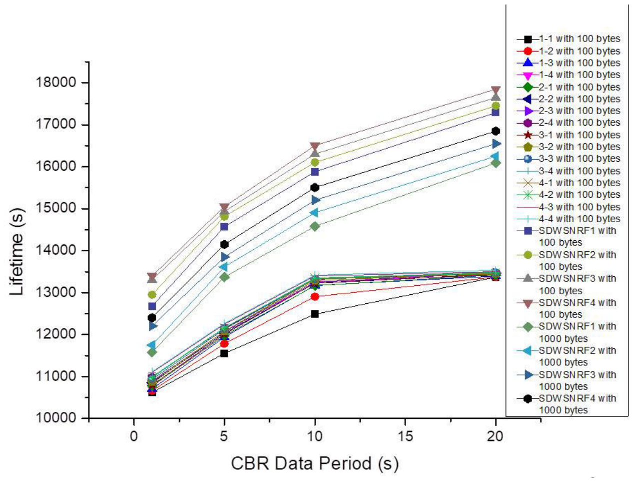

FIGURE 9. Network performance comparison with 100 and 1000 bytes data rate.

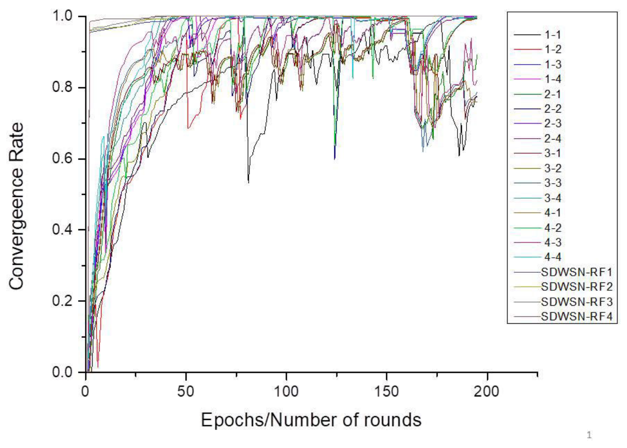

FIGURE 10. Comparison of RL-based techniques for WSN and RL-based SDWSN network convergence rate.

rates of 100 and 1000 bytes, as shown in Figure 9. Even the performance of RL-based SDWSN with the data rate of 1000 bytes is better than RL-based different techniques with a data rate of 100 bytes in terms of network lifetime because the centralized controller in SDWSN observes the network globally. In SDWSN, the SDN controller is also intelligent and learns from previous actions and selects the best path/RT. In SDWSN, the control traffic is also reduced as compared to RL-based different techniques, where each node learns the routing path to select the next forwarder through the information of neighboring nodes. The RL-based SDWSN reduces the network's energy consumption and improves the network lifetime due to the low control traffic.

6) CONVERGENCE RATE COMPARISON OF RL-BASED TECHNIQUES FOR WSN AND RL-SDWSN

In this section, the convergence rate is compared between RL-based techniques for WSN and RL-based SDWSN. RL-based SDWSN can quickly establish the routing paths due to the centralized controller, which manages the network topology and sends the routing table to all the underlying network. However, in the RL-based network, each node finds the forwarder node locally and makes several attempts to find the best forwarder, resulting in the performance degradation of the convergence rate.

The experimental results are shown in Figure 10. It is seen that the RL-based SDWSN converges faster than the RL-based techniques for WSN, which needs more iterations for the convergence. Since SDWSN takes the routing decisions through the centralized SDN controller that takes actions based on the intelligent SDN controller. From the graphical representation, it can be seen that the SDWSN-RF4 has a faster convergence rate among all the proposed techniques, and the worst is the combination of 1-1, which is the RL-based WSN technique.

VI. CONCLUSION

Routing plays an important role in WSNs energy optimization. Traditional routing protocols are not developed to meet the specific WSN requirements. Therefore, the combination of both SDN and RL can be a promising solution to enhance network performance. In this paper, SDN controller uses RL to learn network behavior and takes action based on the previous reward. SDN controller generates the routing table obtained from the spanning tree protocol (STP). STP makes a list of routing paths (also known as RT) in three ways as shown in the given graph G: the first is used to generate all possible routing paths, the second generates the MST (as a routing path) through distance, and the third one is used to generate MST according to the number of hops. SDN controller selects one RT from the list and broadcasts it to all nodes where each node follows the RT and sends the node status data to the controller. To optimize the routing in SDWSN, RL is used that learns from the previous actions. Against each action, the controller receives the reward in terms of $EPLT$ loss. Four different reward functions are proposed. The agent calculates the reward through the selected reward function and selects the next action (RT selection from the list). To get the reward, $EPLT$ and $ENLT$ are used. In the proposed reward functions, the first one gives the global $EPLT$ loss. The second reward function determines the average loss of the $EPLT$, and the third reward function uses the weighted parameter (i.e., the distance of each node to sink) in global. Though, the average loss of $EPLT$ is determined by a fourth reward function.

In SDWSN, the controller controls the whole network. Due to centralized control, each node does not need to share its control data with its neighboring nodes. Each node receives the control data (RT) from the controller to globally view the whole network and provides an efficient routing solution by learning network behavior through Q-learning. These results are compared with RL-based WSN techniques results. The experimental results show that RL-based SDWSN outperforms in terms of network lifetime and improving 23% to 30% as compared to RL-based WSN. It gives a fast network convergence rate as compared to RL-based WSN.

ACKNOWLEDGMENT

The authors thankful to T2RS team and IRIT management who provide them the equipment for the experimental work.

REFERENCES

[1] Y. Zhang, M. Qiu, C.-W. Tsai, M. M. Hassan, and A. Alamri, "Health-CPS: Healthcare cyber-physical system assisted by cloud and big data," *IEEE Syst. J.*, vol. 11, no. 1, pp. 88–95, Mar. 2017.

[2] M. U. Younus, S. U. Islam, I. Ali, S. Khan, and M. K. Khan, "A survey on software defined networking enabled smart buildings: Architecture, challenges and use cases," *J. Netw. Comput. Appl.*, vol. 137, pp. 62–77, Jul. 2019.

[3] K. M. Al-Aubidy, A. M. Derbas, and A. W. Al-Mutairi, "Real-time patient health monitoring and alarming using wireless-sensor-network," in *Proc. 13th Int. Multi-Conf. Syst., Signals Devices (SSD)*, Mar. 2016, pp. 416–423.

[4] M. Usman, "Analysis of the impact of different parameter settings on wireless sensor network lifetime," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 3, pp. 16–21, 2018.

[5] K. Sohraby, D. Minoli, and T. Znati, *Wireless Sensor Networks: Technology, Protocols, and Applications*. Hoboken, NJ, USA: Wiley, 2007.

[6] N. P. Mahalik, *Sensor Networks and Configuration*. Berlin, Germany: Springer, 2007.

[7] M. U. Younus, S. U. Islam, and S. W. Kim, "Proposition and real-time implementation of an energy-aware routing protocol for a software defined wireless sensor network," *Sensors*, vol. 19, no. 12, p. 2739, Jun. 2019.

[8] Y. Zhao, Y. Li, X. Zhang, G. Geng, W. Zhang, and Y. Sun, "A survey of networking applications applying the software defined networking concept based on machine learning," *IEEE Access*, vol. 7, pp. 385–405, 2019.

[9] M. L. Littman, "Reinforcement learning improves behaviour from evaluative feedback," *Nature*, vol. 521, no. 7553, pp. 445–451, May 2015.

[10] R. C. Shah and J. M. Rabaey, "Energy aware routing for low energy ad hoc sensor networks," in *Proc. IEEE Wireless Commun. Netw. Conf. Rec.*, vol. 1, Mar. 2002, pp. 350–355.

[11] S. Yessad, N. Tazarart, L. Bakli, L. Medjkoune-Bouallouche, and D. Aissani, "Balanced energy efficient routing protocol for WSN," in *Proc. Int. Conf. Commun. Inf. Technol. (ICCIT)*, Jun. 2012, pp. 326–330.

[12] M. U. Younus, "Contribution to energy optimization in wsn: Routingbased on RL and SDN oriented routing," Ph.D. dissertation, Ecole Doctorale Math., Informatique, Télécommun. Toulouse (MITT), Univ. Paul Sabatier-Toulouse III, Toulouse, France, 2020.

[13] R. Huang, X. Chu, J. Zhang, and Y. H. Hu, "Energy-efficient monitoring in software defined wireless sensor networks using reinforcement learning: A prototype," *Int. J. Distrib. Sensor Netw.*, vol. 11, no. 10, pp. 360–428, 2015.

[14] H. Mostafaei and M. S. Obaidat, "Learning automaton-based self-protection algorithm for wireless sensor networks," *IET Netw.*, vol. 7, no. 5, pp. 353–361, Sep. 2018.

[15] S. Kim, J. Son, A. Talukder, and C. S. Hong, "Congestion prevention mechanism based on Q-leaning for efficient routing in SDN," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2016, pp. 124–128.

[16] G. Stampa, M. Arias, D. Sanchez-Charles, V. Muntes-Mulero, and A. Cabellos, "A deep-reinforcement learning approach for software-defined networking routing optimization," 2017, *arXiv:1709.07080*. [Online]. Available: http://arxiv.org/abs/1709.07080

[17] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *Proc. 1st Int. ICST Conf. Simulation Tools Techn. Commun. Netw. Syst.*, 2008, p. 60.

[18] C. Yu, J. Lan, Z. Guo, and Y. Hu, "DROM: Optimizing the routing in software-defined networks with deep reinforcement learning," *IEEE Access*, vol. 6, pp. 533–539, 2018.

[19] J. Chavula, M. Densmore, and H. Suleman, "Using SDN and reinforcement learning for traffic engineering in UbuntuNet alliance," in *Proc. Int. Conf. Adv. Comput. Commun. Eng. (ICACCE)*, Nov. 2016, pp. 349–355.

[20] Y. Zuo, Y. Wu, G. Min, and L. Cui, "Learning-based network path planning for traffic engineering," *Future Gener. Comput. Syst.*, vol. 92, pp. 59–67, Mar. 2019.

[21] A. Al-Jawad, P. Shah, O. Gemikonakli, and R. Trestian, "LearnQoS: A learning approach for optimizing QoS over multimedia-based SDNs," in *Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcast. (BMSB)*, Jun. 2018, pp. 1–6.

[22] S. Sendra, A. Rego, J. Lloret, J. M. Jimenez, and O. Romero, "Including artificial intelligence in a routing protocol using software defined networks," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2017, pp. 670–674.

[23] S.-C. Lin, I. F. Akyildiz, P. Wang, and M. Luo, "QoS-aware adaptive routing in multi-layer hierarchical software defined networks: A reinforcement learning approach," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, Jun. 2016, pp. 25–33.

[24] C. Fang, C. Cheng, Z. Tang, and C. Li, "Research on routing algorithm based on reinforcement learning in SDN," *J. Phys., Conf. Ser.*, vol. 1284, no. 1, 2019, Art. no. 012503.

[25] Z. Zhang, L. Ma, K. K. Leung, L. Tassiulas, and J. Tucker, "Q-placement: Reinforcement-Learning-Based service placement in software-defined networks," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2018, pp. 1527–1532.

[26] T. Uzakgider, C. Cetinkaya, and M. Sayit, "Learning-based approach for layered adaptive video streaming over SDN," *Comput. Netw.*, vol. 92, pp. 357–368, Dec. 2015.

[27] Y. Geng, "Mind: A machine learning-based SDN control paradigm," in *Proc. Open Netw. Summit*, Santa Clara, CA, USA, 2016, pp. 1–16.

[28] Z. Zhang, L. Ma, K. Poularakis, K. K. Leung, and L. Wu, "DQ scheduler: Deep reinforcement learning based controller synchronization in distributed SDN," in *Proc. ICC-IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.

[29] Y. Han, B. I. Rubinstein, T. Abraham, T. Alpcan, O. De Vel, S. Erfani, D. Hubczenko, C. Leckie, and P. Montague, "Reinforcement learning for autonomous defence in software-defined networking," in *Proc. Int. Conf. Decis. Game Theory Secur.* Seattle, WA, USA: Springer, 2018, pp. 145–165.

[30] Z. Zhang, L. Ma, K. Poularakis, K. K. Leung, J. Tucker, and A. Swami, "MACS: Deep reinforcement learning based SDN controller synchronization policy design," in *Proc. IEEE 27th Int. Conf. Netw. Protocols (ICNP)*, Oct. 2019, pp. 1–11.

[31] V. Huang, G. Chen, and Q. Fu, "Effective scheduling function design in SDN through deep reinforcement learning," in *Proc. ICC-IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–7.

[32] M. Latah and L. Toker, "Artificial intelligence enabled software-defined networking: A comprehensive overview," *IET Netw.*, vol. 8, no. 2, pp. 79–99, Mar. 2019.

[33] O. G. Matlou and A. M. Abu-Mahfouz, "Utilising artificial intelligence in software defined wireless sensor network," in *Proc. IECON-43rd Annu. Conf. IEEE Ind. Electron. Soc.*, Oct. 2017, pp. 6131–6136.

[34] M. Chakraborty, S. Chowdhury, J. Chakraborty, R. Mehera, and R. K. Pal, "Algorithms for generating all possible spanning trees of a simple undirected connected graph: An extensive review," *Complex Intell. Syst.*, vol. 5, no. 3, pp. 265–281, Oct. 2019.

MUHAMMAD USMAN YOUNUS (Member, IEEE) received the master's degree in engineering from the University of Engineering and Technology Lahore, Pakistan, in 2014, and the Ph.D. degree in engineering from the University of Toulouse (III) Paul Sabatier, France, in 2020. His research interests include wireless sensor networks, software defined networking, energy optimization, wireless communication, and machine learning. He has more than 15 international journal and conference publications. He is a member of PEC and so on and a reviewer of some journals and conferences.

MUHAMMAD KHURRAM KHAN (Senior Member, IEEE) is currently working as a Full Professor with the Center of Excellence in Information Assurance, King Saud University, Saudi Arabia. He has edited seven books and proceedings published by Springer-Verlag and IEEE. He has published more than 370 papers in international journals and conferences. He is an inventor of several U.S./PCT patents. His current research interests include cybersecurity, biometrics, multimedia security, and digital authentication. He is a Fellow of the IET, U.K., BCS, U.K., the FTRA, South Korea, a member of the IEEE Technical Committee on Security and Privacy, and a member of the IEEE Cybersecurity Community. He is the Editor-in-Chief of a well-reputed journal *Telecommunication Systems* (Springer). He is a full-time Editor/an Associate Editor for several international journals/magazines, including IEEE Communications Surveys and Tutorials, *IEEE Communications Magazine*, IEEE Internet of Things Journal, IEEE Transactions on Consumer Electronics, *Journal of Network and Computer Applications* (Elsevier), IEEE Access, *Security and Communication Networks*, and *IEEE Consumer Electronics Magazine*.

MUHAMMAD RIZWAN ANJUM received the B.Engg. degree in electronic engineering and the M.Engg. degree in telecommunication and control engineering from Mehran UET Jamshoro, Pakistan, in 2007 and 2011, respectively, and the Ph.D. degree in information and communication engineering from the Beijing Institute of Technology, Beijing, in 2015. Since 2008, he has been with the Department of Electronic Engineering with The Islamia University of Bahawalpur, where he is currently working as an Assistant Professor. He has more than 30 international conferences and journal publications. His research interests include wireless communication and information sciences. He is member of PEC, IEEEP, IEP, IJPE, UACEE, IACSIT, ICCTD, IACSIT, IAENG, and so on and a reviewer of several journals and conferences.

SHARJEEL AFRIDI (Senior Member, IEEE) received the Bachelor of Engineering degree from Mehran UET, Jamshoro, Pakistan, in 2007, and the M.Sc. and Ph.D. degrees from the University of Leeds, U.K., in 2012 and 2018, respectively. He has worked as a Research Associate with the Institute of Information and Communication, University of Sindh, from October 2007 to July 2009. He has been working as an Associate Professor with the Department of Electrical Engineering, Sukkur IBA University, since July 2009. His research interests include mobile communication systems and microwave filters and measurements. He is a member of various professional bodies, including the Pakistan Engineering Council, the IEEE Microwave Theory and Techniques Society, and the European Microwave Association.

ZULFIQAR ALI ARAIN is currently pursuing the Ph.D. degree from the Institute of Information and Communication Engineering, Beijing University of Posts and Telecommunication, Beijing, China. He is also as an Assistant Professor with the Department of Telecommunication Engineering, Mehran University of Engineering and Technology, Jamshoro, Pakistan. His research interests include energy-aware networking protocols, wireless networking, multimedia communications, and next-generation Internet technologies.

ABDUL ALEEM JAMALI received the Bachelor of Engineering degree in electronic engineering and the Post Graduate Diploma (PGD) degree in telecommunication and control engineering from the Mehran University of Engineering and Technology (MUET), Pakistan, in 2006 and 2008, respectively, the M.Sc. degree in electrical communication engineering (ECE) from the University of Kassel, Germany, in 2010, and the Ph.D. degree from the University of Kassel, Germany, in 2015. His research interests include computational electromagnetic, modeling of nanophotonics, microwave, and optical antennas. He received several awards, including the Young Scientist Award from the International Union of Radio Science (URSI).

• • •