Q-LEARNING BASED ROUTING PROTOCOL TO ENHANCE NETWORK LIFETIME IN WSNS

Arunita Kundaliya and D.K. Lobiyal

School of Computer and Systems Sciences,
Jawaharlal Nehru University, New Delhi, India

ABSTRACT

In resource constraint Wireless Sensor Networks (WSNs), enhancement of network lifetime has been one of the significantly challenging issues for the researchers. Researchers have been exploiting machine learning techniques, in particular reinforcement learning, to achieve efficient solutions in the domain of WSN. The objective of this paper is to apply Q-learning, a reinforcement learning technique, to enhance the lifetime of the network, by developing distributed routing protocols. Q-learning is an attractive choice for routing due to its low computational requirements and additional memory demands. To facilitate an agent running at each node to take an optimal action, the approach considers node's residual energy, hop length to sink and transmission power. The parameters, residual energy and hop length, are used to calculate the Q-value, which in turn is used to decide the optimal next-hop for routing. The proposed protocols' performance is evaluated through NS3 simulations, and compared with AODV protocol in terms of network lifetime, throughput and end-to-end delay.

KEYWORDS

WSNs, Reinforcement learning, Q-learning, Network lifetime, QLRP, QLTPC.

1. INTRODUCTION

The technological innovations in the field of chip design, wireless technology, and embedded electronics in recent times have enabled designing of economical tiny devices, which are capable of computing, sensing, and communicating. These devices or sensor nodes which are composed of an embedded CPU, one or more sensors for sensing and a low power radio for monitoring environmental parameters with limited battery capacity are used to design Wireless Sensor Networks (WSNs). IoT, healthcare, surveillance etc. [12] are some of the applications of WSNs. The sensor nodes have limited resources like memory, bandwidth, CPU processing power, communication range and power supply which typically cannot be recharged once deployed. Since sensor nodes are subject to energy constraints, enhancement in lifetime of network is important for proliferation of WSNs in different applications. Therefore, balancing energy consumption of nodes or saving power through reduction in wastage of energy among nodes is crucial for enhancing the network lifetime [12]. Within the network, a node spends major portion of its energy on communication of packets which may be control or data packets, in computation of optimal routes and for power adjustments. Various measures have been proposed in the past to judiciously use available power of a node. They range from designing low-power hardware to energy efficient protocols and solutions at various protocol layers [13-21] viz., physical, data link, network, and transport. Over a period of time, researchers have successfully been adopting machine learning paradigms, such as reinforcement learning to address the challenges of inherent design constraints of wireless sensor networks, and maximizing utilization of available resources. Each sensor node in the network faces a decision problem of selecting the next hop node, to forward data packet to sink. Routing decisions at intermediate nodes depend on local information

which is stochastic in nature. It is interesting to understand, how routing in WSNs with incomplete information about working environment can be mapped into a sequential decision making problem.

In this paper, methods to enhance network lifetime are proposed at the network layer. The proposed methods are formulated into routing protocols that use node's local information such as residual energy, and hop distance to decide next hop for data forwarding. Out of the two proposed methods, one method investigates the effect of addition of one more decision making parameter i.e., transmission power, on lifetime, throughput and delay in network. It is evident that coverage area of a node can be influenced by variation in transmission power. A change in transmission power level leads to variation in transmission range of a node. In the proposed work, separate routing protocols namely, Q-Learning based routing protocol (QLRP) and Q-Learning based transmission power control protocol (QLTPC), are developed. QLRP is designed using residual energy and hop length and QLTPC is designed using residual energy, transmission power and hop length parameters in making routing decisions. Both protocols use Q-learning, a form of reinforcement learning technique for selection of optimal routing path. Factors such as residual energy of sensor node and hop distance to sink node are used to calculate Q-value of neighboring nodes. This information in turn helps in deciding the next hop node in a routing path. We have used NS3 simulations for performance comparison of proposed protocols with AODV protocol.

Henceforth, a brief introduction of Reinforcement Learning (RL), Q-learning and their usage in the field of WSNs is given in section 2. Section 3 explains the proposed work- QLRP, QLTPC protocols and section 4, demonstrates simulation and result analysis. Section 5, concludes simulation results obtained for different network metrics.

2. RELATED WORK

Reinforcement learning (RL) has been extensively used in wireless networks for optimizing various network parameters such as network lifetime, delay, energy consumption, link quality estimation etc. Various techniques of RL, which are extensively used in WSNs has been discussed in literature. The authors in [1] used Q-Learning technique for allocation of communication channel among independent cognitive nodes in a wireless network. Advantage of this technique is that it requires smaller memory, bandwidth and low computational power. The algorithm is highly scalable and converges faster when correct data rates are selected. Emilio Ancillotti et.al. [2], used reinforcement learning for link-quality estimation in Low Power and Lossy Networks (RPL). Their strategy aims at accurate measurement of link quality with minimized overheads through active probing and reducing wastage in energy. The reward or utility function is dependent on "RSSI (Received Signal Strength Indicator)" and "ETX (link quality)" values. The authors in [3] proposed a method that maximizes the network lifetime of WSN (in intrusion detection application) by introducing a sleep-wake up schedule. The schedule helps in timing sleep of each sensor node in the network. Further, an on-policy Q-learning algorithm proposed for this problem provides solution that models it as a continuous state-action space POMDP. The algorithm operates in two timescales. Faster timescale is used to update the estimates of policy gradient, while slower timescale is used to update the Q-value through on-policy temporal difference method. Authors in [4] applied reinforcement learning algorithm at MAC layer with the aim, to increase the lifetime, and decrease the latency of a Wireless Sensor Networks. Each node has an agent which learns an optimal sleep schedule to achieve its goal. The authors in [5] proposed a new MAC protocol based on Q-Learning and slotted ALOHA. Scheduling of slots for transmission is done in a distributed manner using Q-values. Each node has an agent which is rewarded for a successful transmission of packet. Al-Rawiet. al. [6] have summarized how different RL models such as, MDP, POMDP, and multi-agent reinforcement

learning can be used for routing in various wireless networks. The article also provides an insight to how various routing problems and challenges have been addressed using RL in distributed wireless networks. Boyan et.al. [7], introduced a basic Q-learning based distributed routing protocol named "Q-Routing" to improve packet delivery rate and minimize packet delay to sink. Q-value is estimated for each neighbor of current node as, the minimum time of delivery to sink from the current node. In [8], RL tasks are modelled as routing of data packets from a single source to multiple sinks. Q-learning approach is used to find minimum cost paths based on hop length from the source to multiple destinations via different neighbors in each discovered path. Here, the state of the agent is defined as a tuple consisting of the desired sink to which packet is intended and complete routing information to reach sink through its neighboring nodes. Action is defined as a possible routing decision. Q-value depicts route cost and is estimated as a sum of hop count to all sinks from a given agent. An upstream node's reward is the cost borne by the downstream node for the requested action. In [9], a self-organizing routing algorithm is proposed, to prolong lifetime of WSNs of coal mine robot. The algorithm uses Q-learning method where Q-value is calculated from parameters such as hop distance from sink, residual energy, and energy consumption of node. Neighboring node with maximum Q-value is selected as next hop. Gustavo Künzel et al., [10] used reinforcement learning model for proposing routing protocol in wireless sensor and actuator network. A global routing agent based on Q-learning is used for selection and weight adjustment meant for optimizing routes, and balancing overall latency and minimizing network lifetime. Here, states of routing agents are modelled as a set of weights assigned to hop length and energy level of the nodes. Change in value of these weights, reflects an action. The agent is rewarded whenever an action-state pair decreases latency and increases network lifetime. To optimize network lifetime in WSN, WenjingGuo et.al. [11], proposed a routing protocol based on reinforcement-learning. The protocol evaluates Q-value to select the next forwarding node. Q-value is calculated based of hop count to sink node, link distance and residual energy. Schemes such as transmit power adjustment and feedback carried by the data packets are used to improve packet delivery and energy consumption of nodes. In [24], authors proposed Q-learning algorithm for scheduling of tasks in nodes, so that energy consumption of nodes in WSNs could be reduced. Here, Q-learning method used support vector machine for approximation of value function and hence resolve the dimensionality problem. In [25], authors proposed a MAC layer protocol for WSN in order to extend lifetime of network. The protocol uses Q-learning to self-adjust duty-cycle of WSN node depending on predicted network traffic and neighboring node's transmission state. In [26], authors used Q-learning at network layer for estimation of distance between a node to holes in the network. This information along with node's residual energy is then used for taking routing decision. The routing protocol aims at reducing energy consumption and enhance network's lifetime. In the proposed work, developed protocols are based on one of the RL technique, called Q-learning. Thus, a brief introduction of reinforcement learning and Q-learning have been given in subsequent sections.

2.1. Reinforcement Learning

Reinforcement learning (RL) [22] is about learning to map situations to actions, so that numerical rewards linked with actions, can be maximized. Thus, RL is a computational technique devised to understand and automate, decision making and goal-driven learning. This involves an interaction between agents with their outside environment. An agent is the decision-maker and the learner. In its course of interaction with the environment, an agent may be in any state. The set of all possible states undertaken by an agent constitute its state space. Actions are agent's choices. Set of all possible actions that an agent can take in a given state, constitute its action space for that state. For each action taken, the received reward forms the basis for comparison and evaluation of choices and helps the agent to make an appropriate choice. Inside an agent everything is controllable and known to the agent. Outside the agent, everything may or may not be completely known and is uncontrolled by the agent. At discrete time steps $= 0, 1, 2 \ldots$, an action chosen by

the agent starts its interaction with the environment. The environment responds to agent for chosen action, through presentation of new situations. The environment gives rewards to an agent which it tends to maximize over time through its selected actions. An agent learns about optimal actions through trial-and-error method by exploring its operating environment. Actions taken by an agent also affects the future state of an agent. Despite uncertainty about its environment, an agent seeks to achieve its long term goal depending on its state and available actions. Figure 1 depicts agent's interaction with its environment, where an agent perceives the current status of the environment through the status of its sensors. An action is selected based on the current policy. Execution of an action influences the environment and results in transition in the state of an agent. A feedback is also provided by environment in the form of reward. Therefore, by focusing on learning from direct interactions between agent and its environment without any external supervision to achieve its long-term goal, makes RL distinct from other computational techniques.

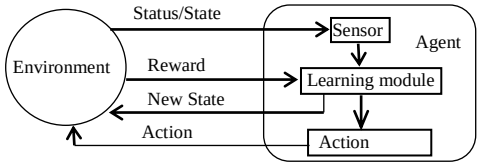

Figure 1: Agent-Environment interaction

Let the operating environment be assumed as stochastic discrete-time, finite-state dynamic system. To define interactions of a learning agent with its environment, RL uses Markovian Decision Process (MDP) framework to describe the states, actions and rewards of the learning agent. The Markovian (or memory less) property of the MDP states that selection of an action by agent at time step n, solely depends on state-action pair at time step $n-1$, and not at time steps $n-2, n-3$ ... i.e., past history. Time steps here, refer to successive stages in decision making. Here, MDP is depicted as tuple consisting of $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{P} \rangle$, in which $\mathcal{S}$ represents set of states, $\mathcal{A}$ represents set of actions undertaken in a given state, $\mathcal{P}$ represents reward function which is described as $\mathcal{P}: \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ where $\mathbb{R}$ represent set of real numbers and $\mathcal{T}$ represents state transition matrix i.e., a matrix that gives probability of transitioning from state s_j to s_{j+1}. It is described as $\mathcal{T}: \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$. Given any state $s_j = s' \in \mathcal{S}$ and action $a \in \mathcal{A}(s')$, the probability of each possible pair of next state $s_{j+1} = \mathrm{s}''$ and reward $r = \mathcal{P}(\mathrm{s}'', a) \in \mathbb{R}$ is defined as:

$$\mathcal{T}(\mathrm{s}'', a, s') = \Pr\{\mathrm{s}'', r | s', \mathrm{a}\} \quad (1)$$

Pr{.} represents probability distribution.

The expected reward for state-action pair can be calculated as:

$$\mathcal{P}(\mathrm{s}'', a) = \mathrm{E}[r | s_j = s', \mathcal{A}(s') = a]$$
$$\mathcal{P}(\mathrm{s}'', a) = \sum_{r \in \mathbb{R}} r \sum_{s' \in \mathcal{S}} \mathcal{T}(\mathrm{s}'', a, s') \quad (2)$$

$E[.]$ represents expected value of a random variable, when agent is in state s' and chooses action a.

Let the expected rewards received by an agent after time step m be:

$$r_{m+1}, r_{m+2}, r_{m+3} \ldots$$

Let TR_m denotes sum of expected rewards received after time step m up to final time step M.

$$TR_m = r_{m+1} + r_{m+2} + r_{m+3} \dots + r_M \quad (3)$$

When $M = \infty$, calculating sum of such large number of values is difficult. So, to determine present values of future rewards, a discounting rate, γ, is used to discount future rewards. An agent tries to choose actions which maximizes sum of discounted rewards an agent receives over future. Now, eqn. (3) becomes

$$TR_m = r_{m+1} + \gamma r_{m+2} + \gamma^2 r_{m+3} \dots$$
$$TR_m = \sum_{n=0}^{\infty} \gamma^n \, r_{m+n+1} \quad (4)$$

Where, $0 \leq \gamma < 1$.

A policy, π, for decision making is defined as a mapping from each state $s' \in \mathcal{S}$ and action $a \in \mathcal{A}(s')$ to the probability $\pi(a|s')$ of choosing action a in state s'. The policies are evaluated through value functions which estimates how good is the performance of chosen action in a given state. In many practical problems, evaluating state-value function for a given policy by agent, becomes difficult due to unknown transition probabilities [23]. RL can be used as an alternative where requirement of transition probabilities is eliminated, and agent gains knowledge via constant interactions with its environment [3][6]. Let $V^{\pi}(s')$ defines value of state s' under policy π.

$$V^{\pi}(s') = E_{\pi}[TR_m | s_m = s']$$
$$= E_{\pi}\left[\sum_{n=0}^{\infty} \gamma^n \, r_{m+n+1} \, | s_m = s'\right]$$
$$= E_{\pi}[\, r_{m+1} + \gamma \textstyle\sum_{n=0}^{\infty} \gamma^n \, r_{m+n+2} \, | s_m = s']$$
$$= \sum_{a \in \mathcal{A}(s'), s' \in \mathcal{S}} \pi(a|s') \sum_{s'' \in \mathcal{S}} \sum_{r \in \mathcal{P}(s'', a)} \mathcal{T}(s'', a, s') \left[r + \gamma E_{\pi} \left[\sum_{n=0}^{\infty} \gamma^n \; \mathcal{P}(s'', a)_{m+n+2} \, | s_{m+1} = s'' \right] \right]$$

Thus, $V^{\pi}(s')$ is represented by eqn. (5)

$$V^{\pi}(s') = \sum_{a \in \mathcal{A}(s'), s' \in \mathcal{S}} \pi(a|s') \sum_{s'' \in \mathcal{S}} \sum_{r \in \mathcal{P}(s'', a)} \mathcal{T}(s'', a, s')[r + \gamma V^{\pi}(s'')] \quad (5)$$

Where, $E_{\pi}[.]$ represents expected value of a random variable, when policy π is followed by the agent.The eqn. (5) is also known as Bellman equation for V^{π} and expresses relation between value of a state and its successive state. RL methods can be broadly categorized into three cases and are briefly described below:

1) When state space is simple, generally single but, action space is large. The agent needs to select an action from the available 'n' actions for which it receives a numerical reward for each chosen action. Goal of the agent is to maximize its reward.
2) When state and action spaces are limited. Here, value function associated with a state-action pair can be approximated by using tables. Such problems can be represented by finite MDP and Bellman equation is used for determining value function of policy. Most common methods used for solving finite MDPs are- temporal difference learning, Monte Carlo methods and dynamic programming. Dynamic programming method needs complete knowledge and accurate modelling of the environment. Monte Carlo methods are simple and model less but, they are not suitable for problems requiring incremental computations. Temporal difference methods like Q-learning and SARSA (state-action-reward-state-action) are model less, completely incremental but, sometimes more difficult to analyse.

3) When state and action spaces are very large. The methods used, do not try to search an optimal policy or optimal value function. These methods are executed for unlimited time, they try to find an approximate optimal policy for value function while limited resources are used in computation.

2.2. Q-Learning

Q-learning, proposed by Watkins, is an off-policy RL method belonging to a class of temporal difference (TD) method. The TD methods do not need a prior model of their environment, instead they learn from their experience to choose an optimal action, in order to achieve their goal. Therefore, in TD method, the learning agent considers only one time-step sample, and then information is bootstrapped that is, an agent learns to update their estimated value based on the parts of other estimates [22]. Q-learning method define, "Q-function" which is an action-value function to estimate Q-values. These Q-values are the expected long-term rewards of an agent for each possible action in its action space. For the MDP, as described in previous section, an action-value function returning Q-value, for a learning agent in state$s_j = s' \in \mathcal{S}$, taking action$a \in \mathcal{A}(s')$, next state$s_{j+1} = \mathrm{s}'' \in \mathcal{S}$, reward $r = \mathcal{P}(\mathrm{s}'', a) \in \mathbb{R}$, discount factor γ and following a policy π can be described as:

$$\mathrm{Q}^{\pi}(\mathrm{s}', \mathrm{a}) = E_{\pi}\left[{}_{j}^{\gamma}R \mid s_j = s', a_j = a\right] \; where \; {}_{j}^{\gamma}R = \sum_{k=0}^{\infty} \gamma^k \, r_{j+k+1} \quad (6)$$

Q-value for every possible state-action pair is maintained in a lookup table called "Q-table". The agent derives its optimal policy, π, on the basis of these Q-values. The policy is associated with the best-action $a' \in \mathcal{A}(s')$ which corresponds to the maximum Q-value for state , $\mathrm{s}'' \in \mathcal{S}$. By using learning rate, $\alpha(0 \leq \alpha \leq 1)$ and discount factor, $\gamma(0 \leq \gamma \leq 1)$, Q-value is modified at time step $k + 1$, by eqn. (7).

$$Q_{k+1}[s', a] \leftarrow Q_k[s', a] + \alpha\left[r + \gamma \, max_{a' \in A(s')} Q_k[\mathrm{s}'', a'] - Q_k[s', a]\right] \quad (7)$$

Higher value of learning rate, α, implies higher speed of learning i.e., giving higher weightage to new estimates of Q-values than older Q-values. If α=1, the estimated new Q-value will be given by eqn. (8):

$$Q_{k+1}[s', a] \leftarrow \alpha\left[r + \gamma \, max_{a' \in A(s')} Q_k[\mathrm{s}'', a']\right] \quad (8)$$

The discount factor, γ, represents weightage given to delayed and discounted rewards. An action is selected based on two strategies- exploitation and exploration. Exploitation enables agent to choose best-possible action $a = \arg max_{a' \in A} Q_t[\mathrm{s}'', a']$ and so, improves network performance. For knowledge improvement agent uses exploration and calculate Q-values for all the state-action pairs. The Q-learning algorithm converges and iteratively searches the state space of dynamic stochastic environment of the agent to provide an optimal solution in the form of action $a \in \mathbb{A}$. Let us consider a tuple which summarizes a simple transition of an agent as $\langle s_0, a_0, r_1, s_1 \rangle$ where, s_0 is the initial state of agent before transition, s_1 is the resulting state after transition, r_1 is the immediate reward received by an agent and a_0 is the action chosen by an agent in state, s_0. Working of Q-learning approach to estimate Q-value, $Q[s, a]$ with learning rate $\alpha(0 \leq \alpha \leq 1)$ and discount factor $\gamma(0 \leq \gamma \leq 1)$is described below:

Algorithm:

Initialize γ, α and $Q[s, a]$ to an arbitrary value
Initialize $s = s_0$
Repeat (for each step in a scenario)
Choose an action, a available for the agent in state s
Execute a and observe agent's new state, s′ and reward r′
Update $Q[s, a] \leftarrow Q[s, a] + \alpha\left[r' + \gamma\, max_{a'}\, Q[s', a'] - \quad Q[s, a]\right]$
$s \leftarrow$ s′
Until s is an end state

In WSNs, Q-learning has been widely used for routing due to its simplicity and model-free approach. It suits well with resource constrained sensor nodes. Also, the inherent broadcasting nature of communication in WSNs, where transmission from a node can be overheard by all neighbors which are one hop away from it, indeed helps in keeping up-to-date estimates of Q-value. In a dynamic stochastic environment, Q-learning approach can prove helpful in identifying efficient routing policies without centralized control and any prior knowledge of network traffic and topology. These distinct features of Q-learning has encouraged us for its use in proposed routing protocols.

3. PROPOSED WORK

In the proposed work two routing protocols based on Q-learning are proposed. In protocol, QLRP, lifetime of network is improved through Q-learning for routing policies. The protocol uses residual energy and hop length as decision parameters. The QLTPC protocol is proposed to study the effect of addition of one more decision parameter, transmission power on lifetime, throughput and delay in the network. The other two parameters are, hop length and residual energy. Q-function uses these parameters for calculating Q-value, which is then stored in a search table called Q-Table. The table is then used for routing decisions and evaluation of alternative routes quality.

3.1. QLRP Protocol

WSNs operates in environment which are dynamic and stochastic in nature. Here, sensor nodes are the agents and the real decision makers, and their environment can encompass other nodes in the network or nodes' properties such as energy level of nodes' battery, position coordinates, modulation levels etc. Actions can be a task as simple as selection of power level for transmission, next hop, modulation, switching radio transceivers etc. In subsequent sections, we describe integration of Q-learning for routing decisions in the protocol, structure of different control packets exchanged between nodes and basic steps of protocol operations.

3.1.1. Integration of Q-Learning in the Protocol

The Q-learning approach has been mapped to the WSN routing protocol as follows:

States: An agent can be either in transmit (T), receive (R) or listen (L) state. When an agent has a packet (data/control) ready for sending, it is in transmit state. When a packet (data/control) is arrived at destination node, agent is in receive state. Otherwise, agent is in listen state. Therefore, state space S of agent is $\{T, R, L\}$.

Action: An action space of an agent consist of {NH} where NH denotes next hop.

Reward: An agent is rewarded only if it has a direct path available to sink.

Q-value: Q-function is used to initialize Q-value. This Q-function is defined as a function of residual energy (RE) of sensor node and path length, which is in terms of number of hops (HP) to the sink. The Q-values are stored in a Q-table and are updated using eqn. (7). The Q-function is defined as:

$$Q(RE, HP) = RE * HP$$

Control packets are used to get current status of the energy level of neighboring agents. To select shortest route containing intermediate agents having higher residual energy, the Q-learning algorithm is slightly modified for this protocol. Instead of selecting state-action pair having maximum value, we choose state-action pair having minimum value. Therefore, a neighbor node who's Q-value is minimum for a given agent is selected as the next hop node. Q-learning method suffers from computational problems which emerges from handling of large state-action space. The problem arises from the need to store Q-value associated with each state-action pair in a Q-table. For a large state-action pair, Q-learning algorithm may become intractable. By maintaining Q-value of those neighbors at one hop distance, Q-table is kept small in size.

Energy model: Energy model used in the protocol is simple. The agent determines its total energy consumed in different states. This information is used to determine residual energy of node as follows:

$$\text{Residual energy} = \text{Initial Energy} - \text{Energy Consumed}$$
$$\text{Energy Consumed} = \text{SV} * \text{C} * \text{T}$$

Where, SV is Supply voltage, C is Amount of current drawn in each state and
T is time duration for which current is drawn

3.1.2. Structure of Control Packets used in the Protocol

In this protocol we have introduced four control packets - Q_REQ, Q_REP, Q_HP, and Q_ERR. Q_REQ packet is used to send request for Q-value, Q_REP is used to send the reply, Q_HP, provides the hop length and Q_ERR notifies the error encountered in the network such as link break. The structure of these packets is described in figure 2(a-d).

Sink's IP address
Source's IP address
Hop Count (HC)

Figure 2(a) Q_REQ

Source's IP address
Hop Count (HC) = 1
Residual Energy

Figure 2(b) Q_HP

Sink's IP address
Source's IP address
Hop Count (HC)
Residual Energy
Lifetime of packet (in msec)

Figure 2(c) Q_REP

Unreachable Sink's IP address
Hop Count (HC) to Sink

Figure 2(d) Q_ERR

Figure 2(a-d) Structure of control packets in QLRP

Q_HP control packets are transmitted at regular interval. If an agent receives Q_HP at regular intervals from its neighbors, links are assumed to be bidirectional. If Q_HP control packet is not heard from a neighbor for a long time, it assumes unidirectional or broken link. If an agent ascertains that the link is unidirectional, it can ask the receiver to send an acknowledgement for the packet received. The flowchart in figure 3(a-b) depicts basic operation of protocol, QLRP.

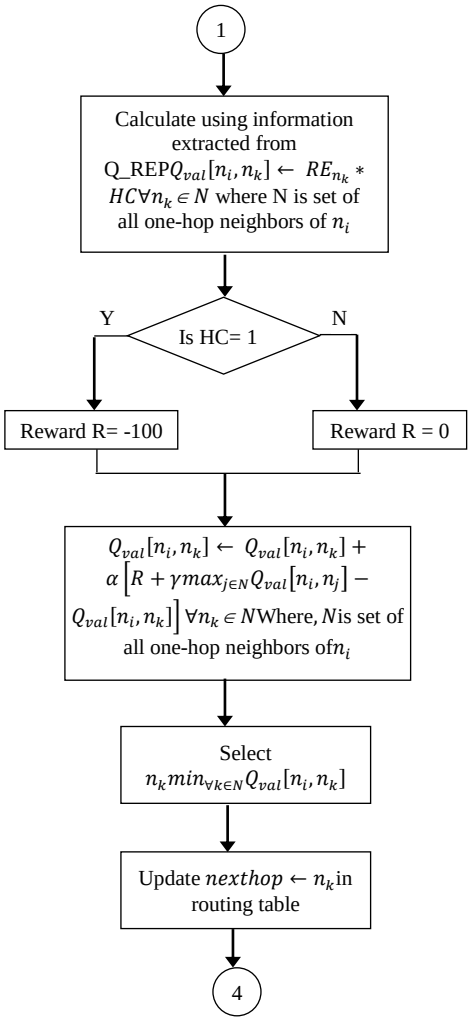

Figure 3(a)

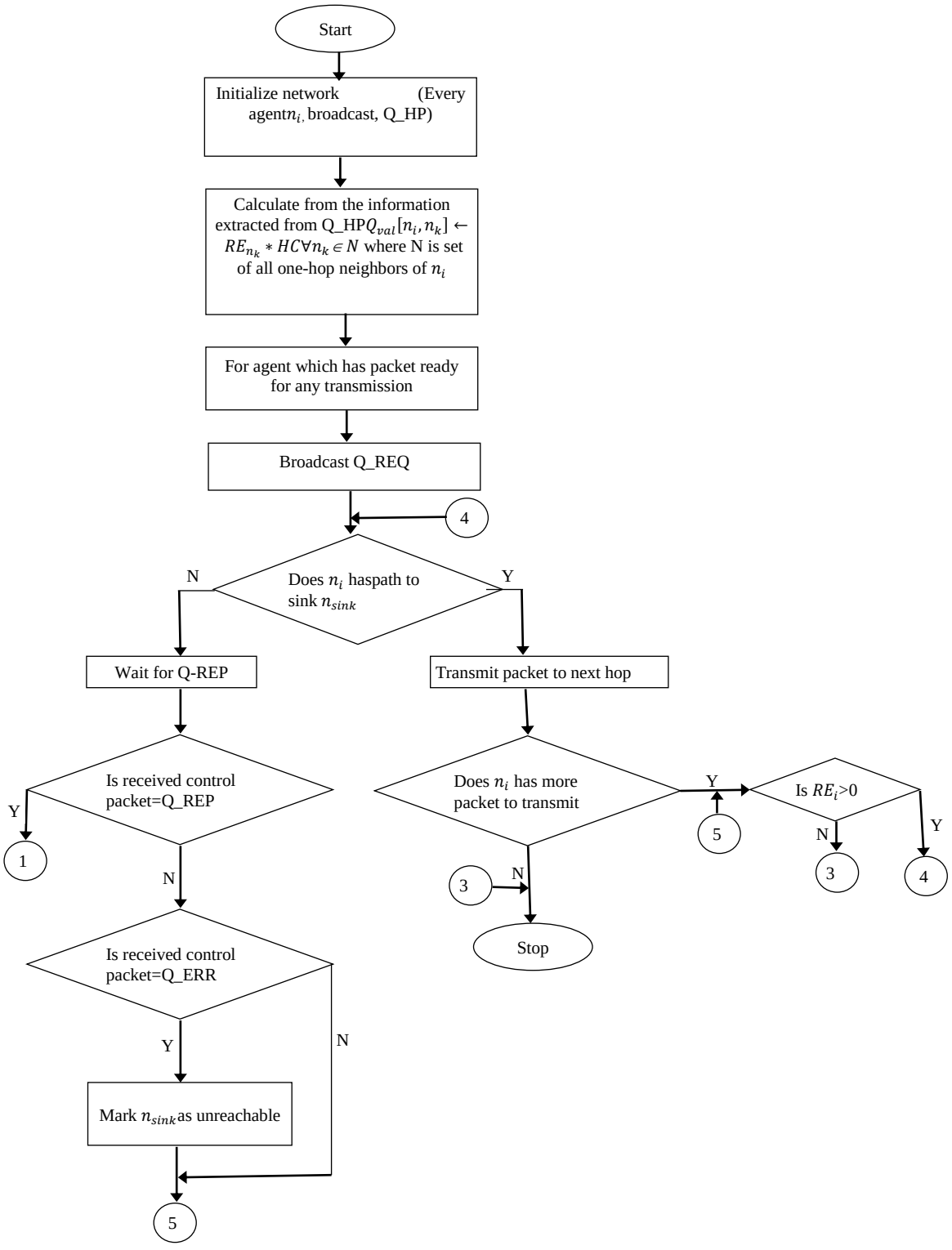

Figure 3(b)

Figure 3(a-b): QLRP protocol

3.1.3. Basic Operation of the Protocol

- Each sensor node in the protocol is an agent. If two agents falls in each other's communication range, they are connected and link between them is bi-directional. When the network is initialised all agents are in listen state, waiting for a packet to arrive.
- An agent broadcasts Q_HP, after initialisation of the network. The packet is received by all neighboring agents at one hop distance. Q_HP contains information about its current energy level i.e. residual energy. Q-function utilizes this information to initialise Q-value of the transmitting agent, in agents receiving the packet.
- Agent switches its state to transmit, when it receives a packet for transmission. The agent broadcasts Q_REQ, which is heard by all one hop neighboring agents.
- An agent on receiving the control packet Q_REQ, rebroadcasts it, if the agent is not the sink node.
- When a sink or an intermediate agent having a route to sink, receive Q_REQ, then that agent generates a control packet Q_REP, which is sent back to the agent from whom Q_REQ was received. In Q_REP, an agent sends information about its current energy level i.e. residual energy.
- An agent on receiving Q_REP calculates new Q-value associated with the neighboring agent by using Q-function described previously. The Q-value is modified through eqn. (7). Routing table is modified, where next-hop is the neighboring agent which has minimum Q-value. All intermediate agents repeat this process, until Q_REP reaches the source.
- The agent at the source node now has complete knowledge about the policy. In other words, the agent has routing path specifying which node to select as the next-hop node as an action. Based on the policy, agent executes the action and transmits the packet to the next-hop.
- In case of communication link failure with the next-hop agent, the agent sends a Q_ERR control packet to the source informing about the link break so that the agent at source explores an alternate route from Q-table, if it is available. In case of no alternative route, it explores new routes by rebroadcasting Q-REQ control packet.
- An agent at source rebroadcasts the Q-REQ control packet, if Q_REP does not arrive at source within a stipulated time.

To explain the basic steps of protocol operation, consider the network in figure 4. In the example, residual energy (RE) of an agent at time t, in the network is indicated. As Q_REQ is broadcasted and forwarded by each agent in the network, for simplicity it is not shown in the figure. Only Q_REP generated by sink, to send response to requested route is shown. Q_REP is send back to only those neighbors from whom route request was received. In the example, S is the source node generating the packet and D is the sink. Link between two nodes indicate that two nodes are in communication range with each other. Let HC denote hop count from sink node. Q-value, calculated through values of reward and Q-function, is denoted as Q here.

Considering α=0.75, γ=0.9, how next hop is selected by the protocol is explained below:

Step 1: When Q_REP is received by node 5, its neighbor set N = {D (HC = 1)}. The Q-values are updated as:

$$\mathrm{Q}[5,\mathrm{D}] = 0.9 * 1 = 0.9$$
$$\mathrm{Q}[5,\mathrm{D}] = 0.9 + 0.75[-100 + 0.9(0.9 - 0.9)] = -74.1$$
$$\mathrm{nexthop} = \mathrm{D}$$

Step 2: When Q_REP is received by node 3, its neighbor set N = {D (HC=1), 5 (HC=2)}. The initial Q-values of the neighbors are:

$$Q[3,\mathrm{D}] = 0.9 \ , Q[3,5] = 0.8 * 2 = 1.6$$
$$Q[3,\mathrm{D}] = 0.9 + 0.75[-100 + 0.9(1.6 - 0.9)] = -73.62$$
$$Q[3,5] = 1.6 + 0.75[\ 0 + 0.9(1.6 - 1.6)] = \ 1.6$$
$$\mathrm{nexthop} = \mathrm{D}$$

Step 3: When Q_REP is received by node 4, its neighbor set N = {5 (HC = 2)}. The Q-values are updated as:

$$Q[4,5] = 0.8 * 2 = 1.6$$
$$Q[4,5] = 1.6 + 0.75[0 + 0.9(1.6 - 1.6)] = \ 1.6$$
$$\mathrm{nexthop} = 5$$

Step 4: When Q_REP is received by node 1, its neighbor set N = {3 (HC = 2)}. The Q-values are updated as:

$$Q[1,3] = 0.8 * 2 = 1.6$$
$$Q[1,3] = 1.6 + 0.75[0 + 0.9(1.6 - 1.6)] = \ 1.6$$
$$\mathrm{nexthop} = 3$$

Step 5: When Q_REP is received by node 2, its neighbor set N = {3 (HC=2), 4 (HC=3)}. The initial Q-values of the neighbors are:

$$Q[2,3] = 0.8 * 2 = 1.6 \ , Q[2,4] = 0.9 * 3 = 2.7$$
$$Q[2,3] = 1.6 + 0.75[0 + 0.9(2.7 - 1.6)] = \ 2.34$$
$$Q[2,4] = 2.7 + 0.75[\ 0 + 0.9(2.7 - 2.7)] = \ 2.7$$
$$\mathrm{nexthop} = 3$$

Step 6: When Q_REP is received by node S, its neighbor set N = {1 (HC=3), 2 (HC=3)}. The initial Q-values of the neighbors are:

$$Q[\mathrm{S},1] = 0.75 * 3 = 2.25 \ , Q[\mathrm{S},2] = 0.7 * 3 = 2.1$$
$$Q[\mathrm{S},1] = 2.25 + 0.75[0 + 0.9(2.25 - 2.25)] = \ 2.25$$
$$Q[\mathrm{S},2] = 2.1 + 0.75[\ 0 + 0.9(2.25 - 2.1)] = \ 2.2$$
$$\mathrm{nexthop} = 1$$

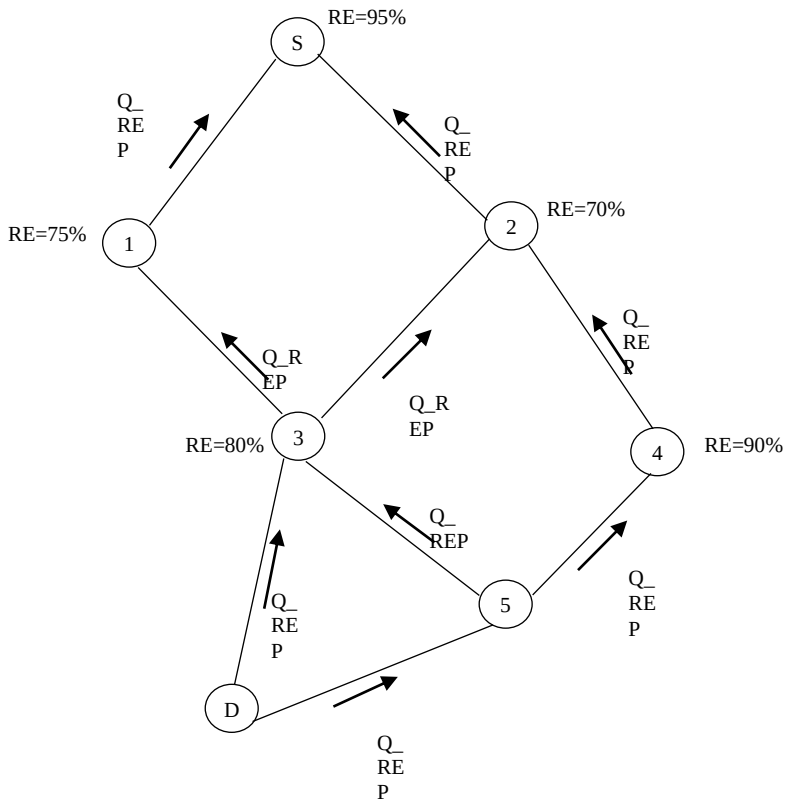

Figure 4: Dissemination of control packet Q_REP between source and destination

3.2. QLTPC Protocol

The objective of this distributed routing protocol is to study the effect of different transmit power levels on the network lifetime, throughput and delay. The power levels for transmission is selected in such a way that nodes which are closer to each other would need less power for transmission. The appropriate selection of power level for transmission helps in saving of power. It avoids unnecessary wastage of power, which is the case when same power level is used regardless of the distance between nodes. The protocol is similar to the QLRP routing protocol as it also uses Q-learning techniques to devise its policy on routing decision. Each sensor node in the protocol is an agent. The agents are connected if the two nodes are in the communication range of each other. To decide next hop it uses same energy model and Q-function to estimate Q-values and update the Q-values by eqn. (7). It differs from QLRP in the way power levels are selected for packet transmission. In QLRP protocol, common power level is used to transmit all the packets, whereas, in QLTPC two power levels are used for data packet transmission. We used distance formula to find distance between the two nodes in routing path. One power level is used to transmit packets to all nodes which are at a distance below a threshold value and another power level for nodes which are above that threshold distance. Control packets used in the protocol are, Q_REQ, Q_REP, Q_HP, and Q_ERR. The structure of Q_REQ, Q_HP, and Q_ERR is same as that of QLRP protocol except for Q_REP. The structure of Q-REP packet is described in figure 5.

Sink's IP address
Source IP address
Hop Count (HC)
Residual Energy
Position coordinates (x, y)
Lifetime of packet (in milliseconds)

Figure 5: Structure of Q_REP

3.2.1. Basic Operations of the Protocol

- On initialization of network, all agents are in listen state and waiting for a packet to arrive.
- An agent broadcasts Q_HP, which is heard by all neighboring agents at one hop distance. The packet contains residual energy information. In receiving agents, this information is used by Q-function to initialise Q-value of the transmitting agent.
- Agent changes its state to transmit, whenever it has a packet for transmission and broadcasts Q_REQ.
- When control packet Q_REQ is received, the agent rebroadcasts it, if it is not the sink node.
- When sink or an intermediate agent who has a route to sink, receives Q_REQ, then that agent generates Q_REP. This packet is sent as response to the agent from whom Q_REQ was received. In Q_REP, an agent sends information about its residual energy and position coordinates. An agent maintains position coordinates along with IP address of its neighbours in a local table. This information is used later for calculation of distance.
- An agent on receiving Q_REP packet calculates new Q-value associated with the neighbouring agent by using Q-function described previously. Q-value is modified by using eqn. (7). The neighbouring agent, who has minimum Q-value is selected as the next-hop node in the routing table. All intermediate agents repeat the process, until Q_REP reaches the source.
- Now, the agent at source has complete knowledge about the policy i.e., routing path specifying which node to select as next-hop as an action. Based on the policy, agent executes the action and finds out its next hop node. It also finds out position coordinates of the next hop from its local table.
- Distance between the two communicating nodes $n_i(x_i, y_i)$ and $n_j(x_j, y_j)$ is calculated as:

$$d = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Based on the calculated distance *d*, a power level is selected to transmit the packet to the next-hop node.

- In case of a link failure an agent sends a Q_ERR control packet to the source node informing about the link break. The agent at source then explores an alternate route from Q-table, if it is available. In case of no alternative route, it explores new routes by rebroadcasting Q-REQ control packet.
- An agent at source rebroadcasts the Q-REQ control packet, if Q_REP does not arrive at the source within a stipulated time.

The flowchart in figure 6(a-b) depicts basic operations of protocol QLTPC.

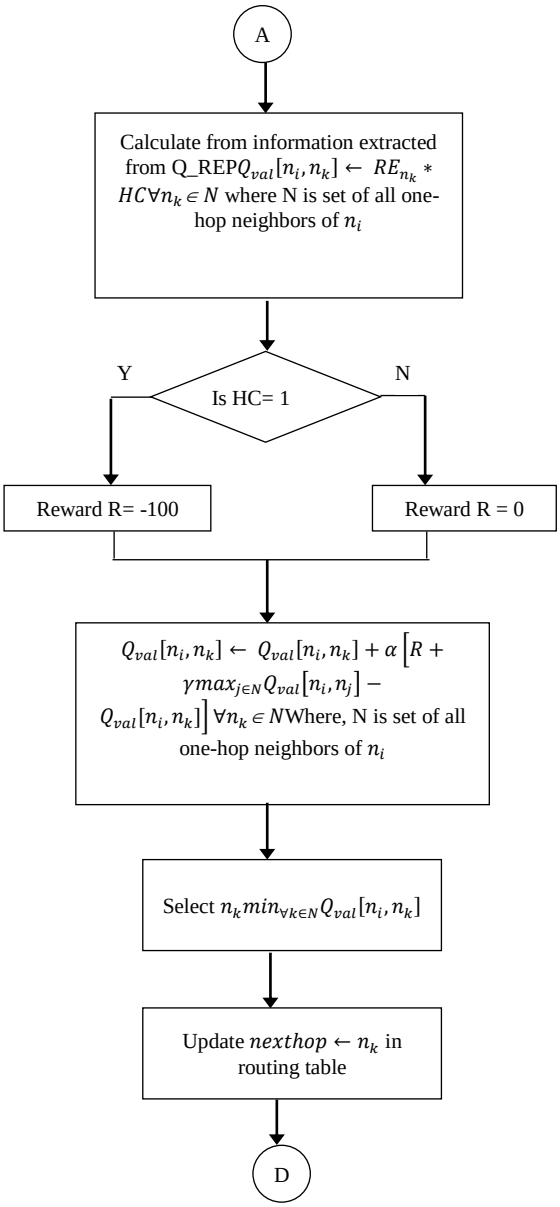
A
Calculate from information extracted from Q_REP$Q_{val}[n_i, n_k] \leftarrow RE_{n_k} * HC \forall n_k \in N$ where N is set of all one-hop neighbors of n_i
Y
Is HC= 1
N
Reward R= -100
Reward R = 0
$Q_{val}[n_i, n_k] \leftarrow Q_{val}[n_i, n_k] + \alpha \left[R + \gamma max_{j \in N} Q_{val}[n_i, n_j] - Q_{val}[n_i, n_k]\right] \forall n_k \in N$Where, N is set of all one-hop neighbors of n_i
Select $n_k min_{\forall k \in N} Q_{val}[n_i, n_k]$
Update $nexthop \leftarrow n_k$ in routing table
D

Figure 6(a)

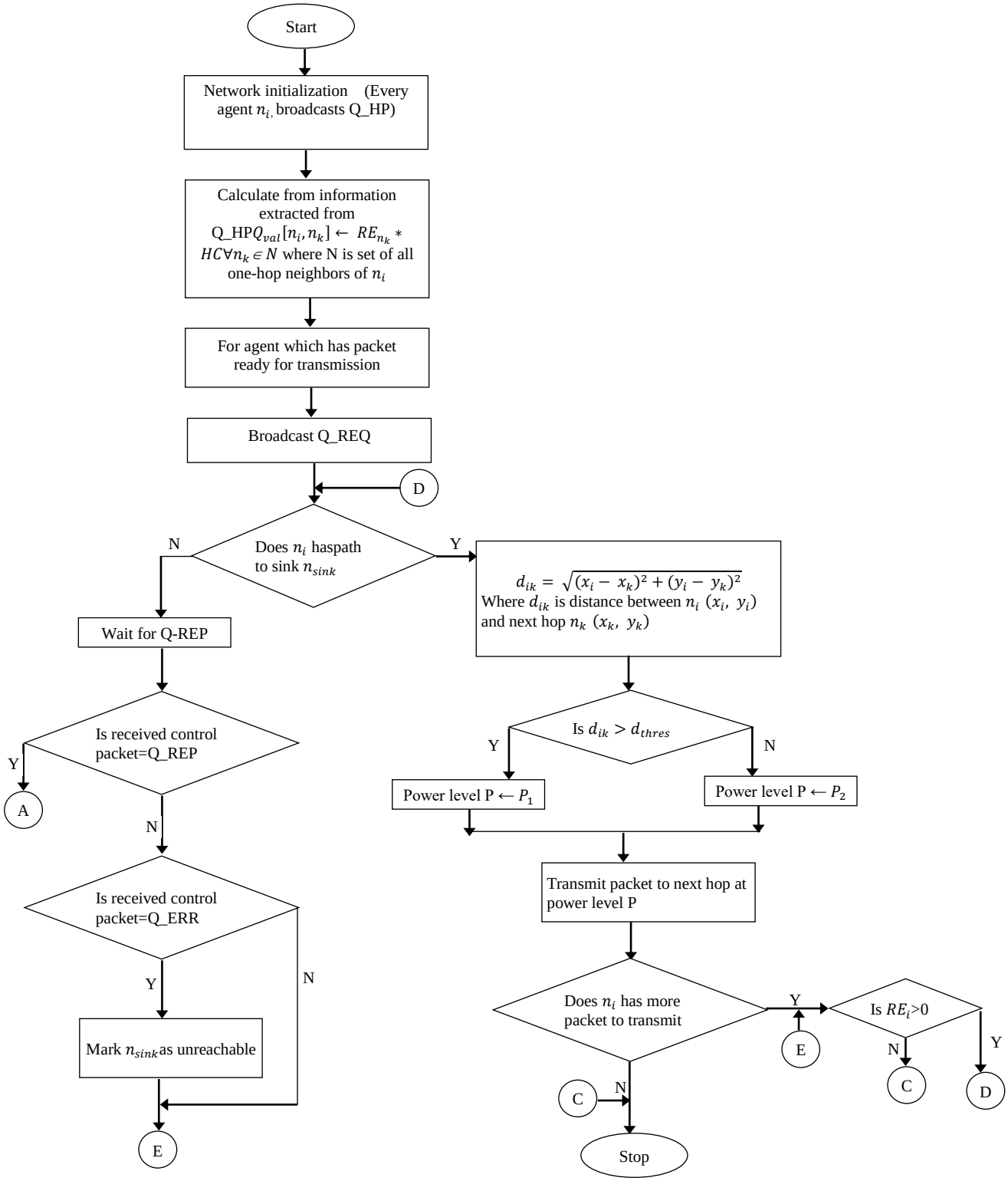

Figure 6(b)

Figure 6(a-b): QLTPC protocol operations

3.3. Advantages of Proposed Protocols

The advantages of proposed protocols are summarized as follows.

1) No central agent to control protocol operations.
2) The routing path is established only, when demanded by an agent with a packet ready for delivery.
3) The sensor nodes (agents) require local information for deciding on next-hop, for packet forwarding.
4) Locally optimizing the links by using Q-learning algorithm.
5) Local routing table at each agent, which are updated through exchange of short control packets between agents.

4. SIMULATION AND RESULTS ANALYSIS

NS3 (ver. 3.27) simulator is used to study routing protocols, QLRP and QLTPC protocol. In the simulation, following performance metrics are used to compare their performance with AODV routing protocol.

- Throughput*:* It is total number of successfully received packets per second at sink.
- End-to-end delay*:* It is time taken by a packet from source to reach the destination, across the network.
- Network Lifetime*:* It is calculated as maximum time period within which the sensor nodes are functional to monitor an event.

In Table 1, different parameters and their corresponding values which are used in simulation of the protocols, are listed below:

Simulation Area(in m)	: 1025x500m
Total nodes in simulation	: 30,40,50,60,70,80,90,100
Total Source-Sink pair	: 10
Mobility model	: Constant Position
Data Packet Size (in Bytes)	: 32, 64, 128
Protocol used at	
(a) MAC layer	: 802.11b
(b) Transport layer	: UDP
Data Rate (in bps)	: 2048
Network Topology	: Random
Radio Propagation Loss Model	:Two ray ground
Initial Node Power(in Joules)	: 100
Power levels used (in dBm)	: 4.0, 4.5, 5.0
Total Simulation Time (s)	: 250s

Table 1: Simulation parameters

To study proposed protocols, network is created where agents are uniformly, randomly distributed in the simulation area. For this purpose, a uniform random variable generator is used to generate the position coordinates of nodes. Nodes do not change their position throughout the simulation period. In the network, ten different source-sink pairs are randomly chosen. Each source node in network, generates packet at the rate of 2048bps. The node can generate packets of variable sizes such as 32, 64 or 128 bytes. Both control and data packets, are transmitted at common power

level 4.5dBm in QLRP protocol. In QLTPC, two power levels are used for transmitting data packets while all control packets are transmitted at common power level. The two proposed protocols are energy-aware protocols which implies that each protocol operation is dependent on node's current energy level. In NS3 [27], energy models are represented through "BasicEnergySource" and "WifiRadioEnergyModel" classes. In simulation, they are used to describe energy source and device energy model respectively. More than one energy source and more than one device energy models could be installed on a node. To draw power from the source, the device energy model has to be connected to the energy source in the node. The "BasicEnergySource" model class provides functions that can keep track of remaining or residual energy. It sends notification to the device energy model in case of complete depletion of energy in the node. The Wifi radio devices have energy model defined in "WifiRadioEnergyModel" class. The "MobilityModel" of NS3 is used to define whether the nodes are mobile or static. It is also used to obtain position co-ordinates of a node, which is needed for QLTPC protocol operation. The WiFi's physical layer parameters and operation intricacies are defined in the class "YansWifiPhy". To compare energy efficiency of protocols, energy source of each node in network is initialized to 100J. In the simulation, to monitor and report packet flows, between a source-sink pair in the network, the "FlowMonitor" model [27-28] of NS3 is used. This model is used to gather end-to-end flow statistics. These statistics can be used through functions defined in class "FlowMonitor". Packets are categorised based on their association with the flow. Take an example of IP flow. This flow contains packets with same protocol, an IP address and port address of both source and sink. For each flow, collected statistics are exported in XML format. Each flow probe assort packets at four points that is, when a packet is send, dropped, forwarded or received by a node. To calculate performance metrics described in eqn. (9) and (10), following attributes [28] from class "FlowMonitor::FlowStats" have been used:

- "timeFirstRxPacket": "It is the time when first packet is received by the sink, in a flow".
- "timeLastRxPacket": "It is the time when last packet is received by the sink, in a flow".
- "delaySum": "It is sum of end-to-end delays, for all received packets which belong to the same flow".
- "rxPackets" : "It is the sum of all the received packets of the sink node in a flow".

$$\text{Average End to end delay (ms)} = \frac{\text{delaySum}}{\text{rxPackets}} \quad (9)$$

$$\text{Avg Throughput (kbps)} = 8 * rxBytes/(TL_{rx} - TF_{rx})/1024 \quad (10)$$

where, TL_{rx} is "$timeLastRxPacket$" , and TF_{rx} is "$timeFirstRxPacket$"

For each network size, simulation is run for ten different seed values of a uniform random variable generator. For each seed value, simulation runtime is of 250s. Average value is calculated from the resultant data obtained after each run, and is considered as the final value of a metric. A comparison in performance of two protocols, QLRP with AODV in terms of metrics throughput, delay and network lifetime are shown in figure 7(a-c), 8(a-c), 9(a-c) respectively. For each metric the two protocols are simulated for different packet sizes and varying node densities.

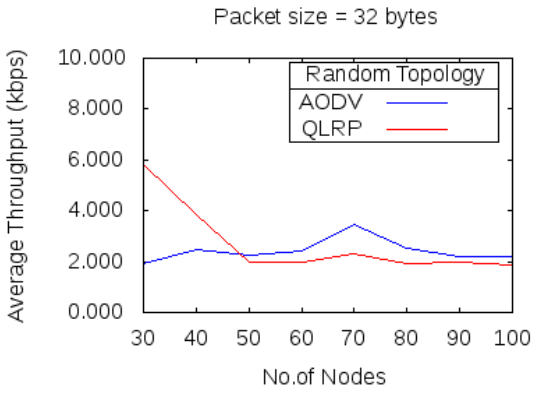

Figure 7(a)

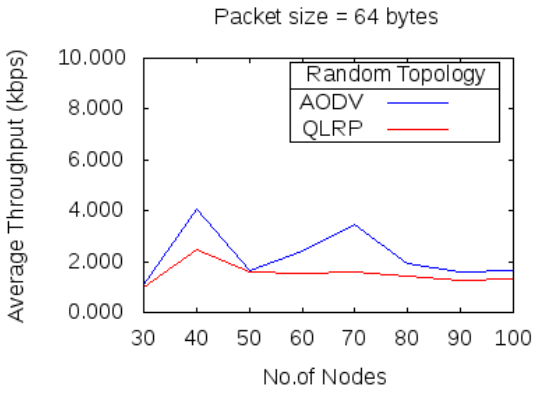

Figure 7(b)

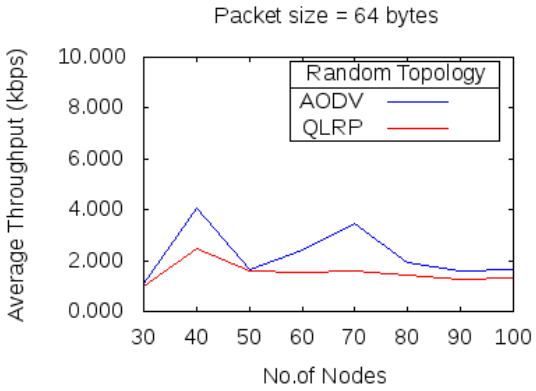

Figure 7(c)

Figure 7(a-c): Average throughput for different packet sizes and nodes

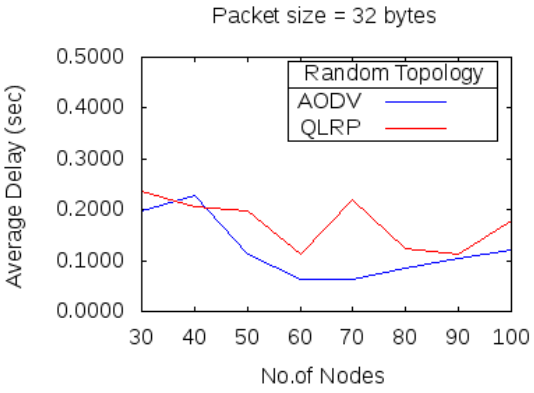

Figure 8(a)

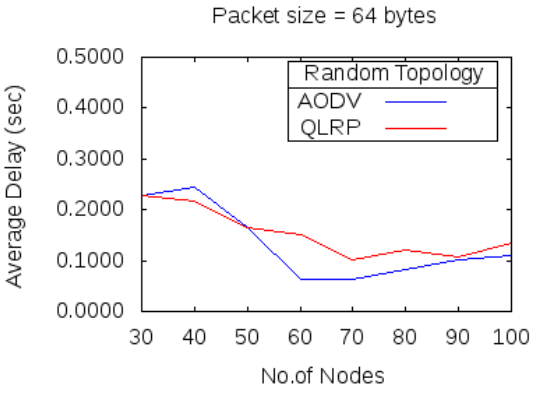

Figure 8(b)

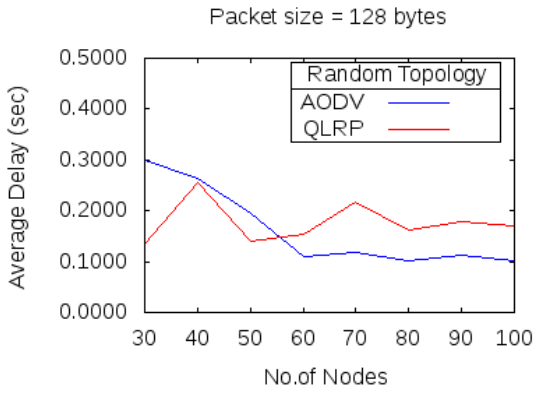

Figure 8(c)

Figure 8(a-c): Average delay for different packet sizes and nodes

The simulation results of the two protocols QLRP and AODV in figure 7(a-c), 8(a-c) have shown that for packet size of 32 bytes, throughput of QLRP protocol in comparison with AODV, is higher for network with low node density. The throughput decreases when number of nodes in the network becomes 50 or more. For the packet size of 64 bytes, throughput is lower for all node densities, while for 128 bytes packet size, throughput is high for network with 30, 40, 60 and 70 nodes and is low for network with 50, 80, 90 and 100 nodes, when compared with AODV. Also, in QLRP protocol, for packet size of 32 bytes, network delay is higher than AODV for all node densities except when number of nodes in the network are 40. For 64 bytes and 128 bytes packet size, delay is lower in QLRP than AODV for all network sizes below 60 and is high as number of nodes in the network increases. When QLRP protocol is compared with AODV in terms of network lifetime in figure 9(a-c), then for 32 bytes packet size network, lifetime in QLRP protocol is higher than AODV for network sizes of 30, 40, 60, 80 and 90 nodes, and for other network sizes of 50, 70, and 100 nodes network lifetime is lower than AODV. For packet size of 64 bytes, network lifetime is lower only for network with 80 nodes and for others it is higher. For 128 bytes packet size, network lifetime is higher than AODV for number of nodes above 30 and below 70 in the network.

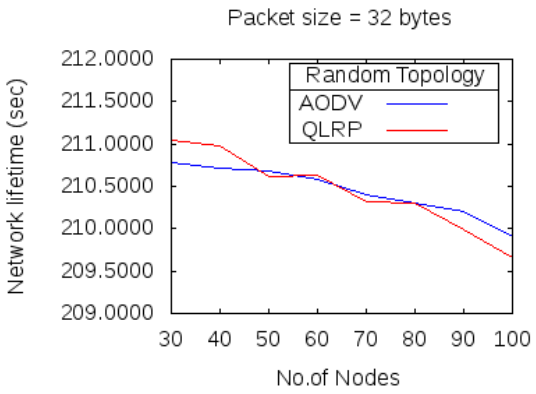

Figure 9(a)

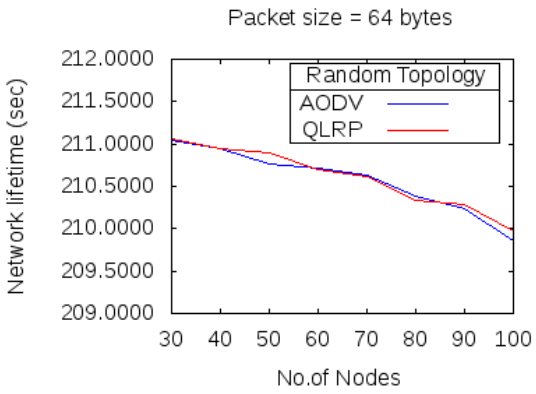

Figure 9(b)

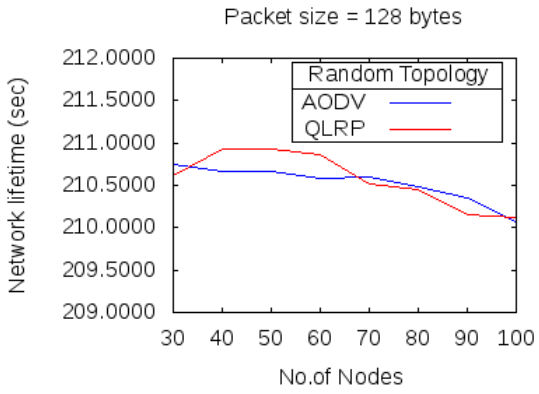

Figure 9(c)

Figure 9(a-c): Network lifetime for different packet sizes and nodes

The QLTPC protocol is developed to study the effect of variation in the transmission power of the sensor node on the performance of the network.

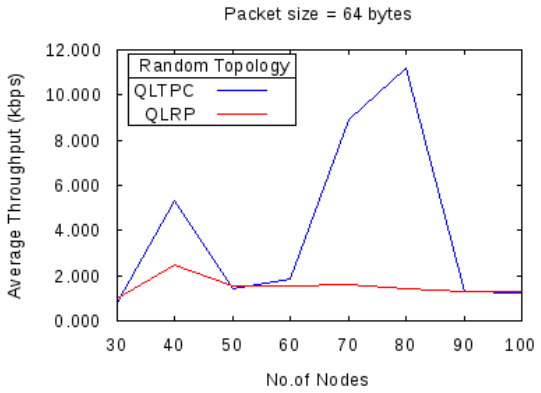

Figure 10(a)

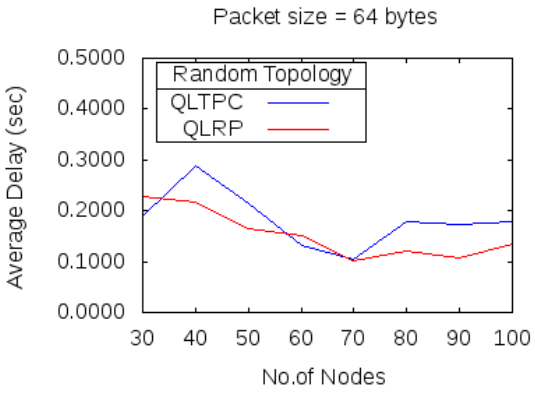

Figure 10(b)

Figure 10(a-b): Average throughput, and delay of QLRP and QLTPC

In this protocol, routing decisions are facilitated by Q-learning technique. The performance of this protocol is compared with the performance of QLRP protocol. Figure 10(a-b) and 11 shows the performance of protocols, QLRP and QLTPC, in terms of throughput, delay and network lifetime for varying packet size and node density. In figure 10(a-b), the simulation results of average throughput and delay for protocol QLTPC and QLRP are shown, for the packet size of 64 bytes. As QLTPC has one additional decision parameter, i.e. transmission power, other than QLRP, in QLTPC, both throughput and delay are higher than QLRP for all network sizes. Delay in QLTPC is lower than QLRP only for network sizes between 60 and 70 nodes. In figure 11, network lifetime of protocols, QLRP and QLTPC is compared. It is observed that, network lifetime in QLTPC has shown little improvement in comparison with QLRP.

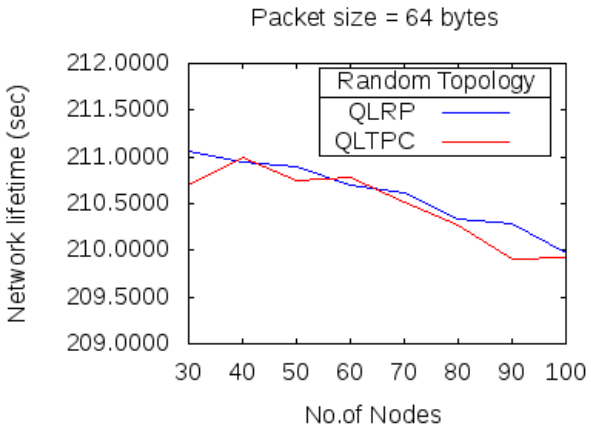

Figure 11: Network lifetime of QLRP and QLTPC protocol

5. CONCLUSION

In the proposed work, routing decisions are taken by agents through Q-learning technique. The agents use residual energy, hop-length to sink and transmission power as decision parameters in protocols. The performance of protocols QLRP, QLTPC and AODV is evaluated w.r.t. metrics such as, throughput, delay and lifetime of network. The metrics are evaluated for different packet sizes and node densities. While comparing protocols, QLRP and AODV, it is found that for packet size of 64 Bytes, Q-learning technique used in QLRP protocol for routing decisions has shown improvement in the network lifetime. Regardless of the packet size, network lifetime has improved. The protocol has also shown improvement in terms of throughput for large packet size (128 bytes). QLTPC protocol, has shown little improvement in network lifetime of network. This protocol has shown more improvement in throughput of the network in comparison with QLRP. Thus, addition of transmission power as decision parameter, other than residual energy and hop distance has improved throughput and network lifetime. Since, energy is also spent in transmitting control packets, in future, we would like to further improve network's operational time by reducing total control packets transmitted i.e., routing overhead, in the network.

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] Hosey, N., Bergin, S., Macaluso, I., &O'Donoghue, D. "Q-learning for cognitive radios." In Proceedings of the China-Ireland Information and Communications Technology Conference: 9780901519672. (2009).

[2] Ancillotti, Emilio, Carlo Vallati, Raffaele Bruno, and Enzo Mingozzi. "A reinforcement learning-based link quality estimation strategy for RPL and its impact on topology management." Computer Communications 112: 1-13. (2017).

[3] Prashanth, L. A., Abhranil Chatterjee, and ShalabhBhatnagar. "Two timescale convergent Q-learning for sleep-scheduling in wireless sensor networks." *Wireless networks* 20, no. 8: 2589-2604. (2014).

[4] Mihaylov, M., Tuyls, K., &Nowé, A. "Decentralized learning in wireless sensor networks". In *International Workshop on Adaptive and Learning Agents*: 60-73. (2009).

[5] Kosunalp, Selahattin, Yi Chu, Paul D. Mitchell, David Grace, and Tim Clarke. "Use of Q-learning approaches for practical medium access control in wireless sensor networks." *Engineering Applications of Artificial Intelligence* 55: 146-154. (2016).

[6] Al-Rawi, Hasan AA, Ming Ann Ng, and Kok-Lim Alvin Yau. "Application of reinforcement learning to routing in distributed wireless networks: a review." *Artificial Intelligence Review* 43, no. 3: 381-416. (2015).

[7] Boyan, J. A., & Littman, M. L. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Advances in neural information processing systems* (pp. 671-678). (1994).

[8] Forster, A., & Murphy, A. L. "FROMS: Feedback routing for optimizing multiple sinks in WSN with reinforcement learning". *3rd International Conference on Intelligent Sensors, Sensor Networks and Information*: 371-376. (2007).

[9] Ma, Xiliang, and Ruiqing Mao. "Method for Effectively Utilizing Node Energy of WSN for Coal Mine Robot." *Journal of Robotics* (2018).

[10] Künzel, Gustavo, Gustavo PedrosoCainelli, Ivan Müller, and Carlos Eduardo Pereira. "Weight adjustments in a routing algorithm for wireless sensor and actuator networks using Q-learning." *IFAC-PapersOnLine* 51, no. 10: 58-63. (2018).

[11] Guo, Wenjing, Cairong Yan, and Ting Lu. "Optimizing the lifetime of wireless sensor networks via reinforcement-learning-based routing." *International Journal of Distributed Sensor Networks* 15, no. 2: 1550147719833541. (2019).

[12] Yetgin, Halil, Kent TszKan Cheung, Mohammed El-Hajjar, and LajosHanzoHanzo. "A survey of network lifetime maximization techniques in wireless sensor networks." *IEEE Communications Surveys & Tutorials* 19, no. 2: 828-854. (2017).

[13] Kawadia, V., & Kumar, P. R. "Principles and protocols for power control in wireless ad hoc networks". *IEEE journal on selected areas in communications*, *23*(1):76-88. (2005).

[14] Raghunathan, V., Schurgers, C., Park, S., & Srivastava, M. B. "Energy-aware wireless microsensor networks". *IEEE Signal processing magazine*, *19*(2):40-50. (2002).

[15] Anastasi, M. Conti, M. Di Francesco, A. Passarella, Energy conservation in wireless sensor networks: A survey, Ad Hoc Networks 7, 537–568. (2009).

[16] Yahya, Bashir, and Jalel Ben-Othman. "Towards a classification of energy aware MAC protocols for wireless sensor networks." *Wireless Communications & Mobile Computing*: 1572-1607.(2009).

[17] Cano, Cristina, et al. "Low energy operation in WSNs: A survey of preamble sampling MAC protocols." *Computer Networks*: 3351-3363. (2011)

[18] Huang, Pei, et al. "The evolution of MAC protocols in wireless sensor networks: A survey." *IEEE communications surveys & tutorials*: 101-120. (2013).

[19] Pantazis, Nikolaos A., Stefanos A. Nikolidakis, and Dimitrios D. Vergados. "Energy-efficient routing protocols in wireless sensor networks: A survey." *IEEE Communications surveys & tutorials*: 551-591. (2013).

[20] Yan, Jingjing, Mengchu Zhou, and Zhijun Ding. "Recent advances in energy-efficient routing protocols for wireless sensor networks: A review." *IEEE Access*: 5673-5686. (2016).

[21] Gholamzadeh, B., & Nabovati, H. "Concepts for designing low power wireless sensor network". World Academy of Science, Engineering and Technology: 559-565. (2008).

[22] Sutton RS, Barto AG, Reinforcement learning: an introduction. MIT Press, Cambridge. (1998).

[23] Liu, Zhenzhen, and ItamarElhanany. "RL-MAC: a reinforcement learning based MAC protocol for wireless sensor networks." *International Journal of Sensor Networks*: 117-124. (2006).

[24] Wei, Zhenchun, et al. "A Q-learning algorithm for task scheduling based on improved SVM in wireless sensor networks." *Computer Networks*: 138-149. (2019).

[25] Savaglio, C., Pace, P., Aloi, G., Liotta, A., &Fortino, G. "Lightweight reinforcement learning for energy efficient communications in wireless sensor networks". *IEEE*: 29355-29364. (2019).

[26] K. Le, T. H. Nguyen, K. Nguyen and P. L. Nguyen, "Exploiting Q-Learning in Extending the Network Lifetime of Wireless Sensor Networks with Holes," IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS), Tianjin, China:602-609. (2019).

[27] https://www.nsnam.org/

[28] Carneiro, Gustavo & Fortuna, Pedro & Ricardo, Manuel. "FlowMonitor - a network monitoring framework for the Network Simulator 3 (NS-3)". 10.4108/ ICST.VALUETOOLS2009.7493. (2009).

AUTHORS

Arunita Kundaliya has done M.Tech. and is currently pursuing Ph.D. from Jawaharlal Nehru University. Her research interest includes Wireless Networks (Ad Hoc and Sensors), Internet of Things (IoT) and Artificial Intelligence.

Dr D.K. Lobiyal is a Professor in the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, Inida. He received his Ph.D in Computer Science in 1996 and Masters in Computer Science in 1991 from Jawaharlal Nehru University, New Delhi. He obtained his Bachelor degree in Computer Science in 1988 from Institute of Engineering and Technology, Lucknow University, India.

His areas of research interest include Wireless Networks (Ad Hoc and Sensors), Natural Language Processing, and Computational Neuroscience. He has published research articles in the international journals of repute including IEEE, ACM Elsevier, Wiley, and Springer. In addition, he has also published more than 80 research articles in the international/national conferences. He is senior member of IEEE society.