Analyzing the energy efficient path in Wireless Sensor Network using Machine Learning

Tirtharaj Sapkota[1], Bobby Sharma[2]

[1] Department of Computer Science and Engineering,
Assam Don Bosco University, Guwahati, India,
Guwahati, India
tirtharaj202@gmail.com

[2] Department of Computer Science and Engineering
Assam Don Bosco University, Guwahati, India,
Guwahati, India
bobby.sharma@dbuniversity.ac.in

Abstract: *As the sensor nodes are energy constrained, an important factor for successful implementation of a Wireless Sensor Network (WSN) is designing energy efficient routing protocols and improving its lifetime. Network life time has been described in many ways such as the time when the network lost its connectivity or the time when the first node gets disconnected. Whatever may be the description, the main focus of many researchers is to design algorithms that enable the network to perform continuously for a longer duration. So, improving the energy efficiency and increasing the network lifetime are the two key issues in WSN routing. Because of the intelligent nature and learning capacity, reinforcement learning (RL) algorithms are very suitable for complex distributed problems such as routing in WSN. RL is a subclass of Machine Learning techniques. It can be used to choose the best forwarding node for transmitting data in multipath routing protocols. A survey has been made in this paper regarding the implementation of RL techniques to solve routing problems in WSN. Also, an algorithm has been proposed which is a modified version of original Directed Diffusion (DD) protocol. The proposed algorithm uses Q-learning technique which is a special class of RL. Also, the significance of balancing the exploration and exploitation rate during path finding in Q-learning has been demonstrated using an experiment implemented in python. The result of the experiment shows that if exploration-exploitation rate is properly balanced, it always yields an optimum value of the reward and thus path found from source to the destination is efficient.*

Keywords: ***Wireless sensor network, Energy efficiency, Network Lifetime, Reinforcement learning, Q-Learning, Exploration vs Exploitation.***

(Article history: Received: 29th September 2021 and accepted 20th December 2021)

I. Introduction

WSN generally consist of thousands of sensor nodes which have very low communication range and thus the position of these nodes should be very close to each other to make the communication still possible [1]. These nodes operate with small batteries which have limited energy to do their task and have low computational ability. The sensor networks are generally deployed in remote and dynamic areas so changing or recharging their batteries to increase the life time of the network is not practical [2]. In a WSN, when a sensor produces some data, then it is required to send the data to destination node. If the destination cannot be reached from the present node, then the present node selects one of its neighbor nodes to carry the data forwarding. The process to select the neighbor node and construct a path from source to destination node is termed as routing.

During routing considerable amount of energy is consumed [3]. To make the sensor network work for a longer duration the energy consumption should be minimized. Therefore, energy efficiency and network lifetime are two issues of research in WSN routing [4]. Energy efficiency of a WSN can be considered as packets received by the destination node divided by the sum of energy consumed. Network life time can be defined as the time when all the sensor nodes have lost their energy and the network as a whole cannot function [5].

In this paper we will introduce algorithms which use RL based technique to determine the optimal path from source to the destination in WSN. RL is a field of machine learning that has been widely used in WSN routing to design algorithms that are energy efficient [6]. Any RL based scenario has two major parts, an agent and environment. The aim is to find a policy that keeps on updating based on the reward which the agent obtains after doing some tasks in the environment until optimum result is obtained [7].

Q-learning is a sub-area of reinforcement learning which is especially used in WSN routing. It is a value-based reinforcement learning algorithm which is used to determine the best action-selection policy according to a Q function. The aim of Q-learning is to maximize the value of Q which generally indicates the goodness of a given action in gaining some future reward [8].

The proposed algorithm is based on Reinforcement learning especially Q-learning. RL is suitable for complex distributed problems like WSN as it can achieve better results. It handles how an agent would perform actions to optimize the long-term reward. In this case each sensor node can be considered as an agent who will learn how to determine the optimum path from source to the destination. A very essential feature of Q-learning is how we define the reward function that updates the Q-value properly [5].

II. ENERGY MODEL

The First order radio model is generally used in calculating the energy consumption in WSN [5][9]. According to this model during transceiving packets by a sensor node, the energy consumed during transmission and reception of packet of data can be represented using expression 1 and 2 respectively.

$$E_{TX}\ (k,d) = E_{elec}\ k + \varepsilon_{amp}\ kd^{m} \qquad (1)$$

$$E_{RX}\ (k) = E_{elec}\ k \qquad (2)$$

Where,
k=length of packet
d= distance to transmit
E_{TX} (k,d) = energy required in transmitting packet of length k to d
E_{RX} (k)= energy required to receive a packet of length k
E_{elec}= Energy required for transceiver circuitry (50 nJ/bit is normally used)
ε_{amp} = Energy required for amplification during transmission (100pJ/m2 is generally used)
m indicates reduction in weight of electromagnetic wave as it travels in space. Its value is 2 when the signal is propagating in free space where as its value is 4 when propagating in relatively lossy environment.

III. REINFORCEMENT LEARNING

RL has become one of the most important types of machine learning technique in today's world. It is a process where an agent is trained to act in some particular situation by doing some actions and seeing the result. RL has agent and environment as major components. An agent sees the existing condition of the environment and finalizes an action according to current policy. After taking the action, agent will obtain some reward from the environment and depending on the reward received the agent modifies its policy. It then goes to the next state. From the next state the agent will use its updated policy to take a new action. In this way the policy keeps on updating. The aim of the agent is to optimize the total reward and it accomplishes this by aggregating the immediate reward and the expected future reward. An RL problem can be structured as a Markov decision Process (MDP) which consist of a 4-tuple (S; A; P; R) Where S indicates set of possible states, A represents set of possible actions, P is the probability of state transition and R symbolizes environmental reward [5][6][7].

IV. Q-Learning

Q-learning is a field of reinforcement learning algorithm that is used to obtain the optimal action-selection policy based on Q function. The Q function produces a value which is used in deciding the next action to take. The aim is to find the maximum value of the function Q. Q stands for quality. It represents how useful a given action is in gaining some future rewards. It can be represented as matrix having states as row and actions as column. The basic steps of Q-learning algorithm are shown in the Figure 1. The Q-table can be updated using the following formula [16]

$$\textbf{Q (state, action) = Reward (state, action) + } \boldsymbol{\gamma}\textbf{*Max [Q (next-state, all action)]}$$

γ is the learning rate i.e. the rate at which the machine learns. The learning rate signifies the speed by which model is adjusted to the scenario. Deciding the learning rate is quite difficult because a very small value may result in a prolonged training process that could get stop in the middle, and a very large value may consequence in learning unacceptable set of weights very early or an uneven training process. The value of Gamma should be adjusted between 0 to1. If it is nearer to zero then the agent may consider only the immediate rewards. If it is nearer to 1, the agent will be inclined towards the future rewards. So the value of Gamma is very critical when designing the algorithm. It can be manually adjusted throughout the training process to improve performance or implemented through some programs. Reward is a scalar quantity received by the agent from the environment when performing an action **a** on state **s**.

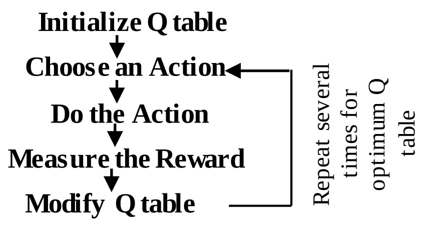

Fig.1. Steps of Q-learning Algorithm

```
Set the value for gamma and rewards in matrix R.
Initialize matrix Q to zero
For each episode:
  Select a random initial state.
  Do until the goal state is reached.
        Select one action for the current state.
        Using this possible action, go to the next state.
        Get maximum Q value for next state
        Compute:
              Q(state, action) = R(state, action)
            + γ *Max[Q(next state, all actions)]
        Set the next state as the current state.
     End Do
 End For
```

Fig.2. Q-Learning Algorithm

The Q-learning algorithm in further detail can be stated as in Figure 2 [16]. Using this algorithm, the agent can learn from knowledge acquired in past. It consists of several episodes. Each episode starts by choosing an initial state randomly and ends when the agent reaches the terminating state. Each episode is called training session. In the beginning the agent in not aware of the environment and therefore all the Q matrix values are zero. The agent explores the environment during each training session which is indicated by reward matrix, obtaining the available rewards on the way before it reaches the terminating state. It is intended towards improving the brain of the agent (which is indicated by Q) using several sessions. The optimized value of Q is directly proportional to the number of training sessions [16].

V. LITERATURE REVIEW

Over the past decade, Machine learning techniques have gained popularity among the researchers of routing protocol in WSN. Reinforcement Learning is one of the types of Machine Learning technique where an agent takes action according to some policy in an environment to get some kind of cumulative reward. This reward helps to modify the policy and then the agent again takes some other actions according to the updated policy. In this way the agent can learn to take best action. In WSN this learning feature of RL can be used to determine paths from source to the sink that are energy efficient. In the following section, we have described few RL-based energy efficient routing protocols in WSN that are found in literature.

The authors in [5] have proposed Reinforcement Learning Based Routing (RLBR) to eliminate the problem of overhead brought about by flooding and route maintenance phase in Energy Aware Routing (EAR) and improve the lifetime of WSN. Here Q-learning technique is used to determine the optimum path from source to the destination. While defining the reward function factors like remaining energy, number of hops, link distance was considered. The simulation results show that compared to other existing protocol, RLBR optimizes the network lifetime in several aspects. The authors in [10] have presented a protocol called MRL: SCSO which describes about self-configuration and self-organization in Unattended WSN. It is a protocol that controls the topology and disseminates the data using multi-agent reinforcement learning technique. Depending on the simulation results, the authors claimed that better Quality of Service compared to CTP (Collect Tree Protocol) is gained with regard to packet delivery ratio, end-to-end delay and throughput. Factors like remaining energy of the node and buffer size were taken to formulate the reward function. In [11] the authors have proposed Feedback routing for optimizing multiple Sink (FROMS) in WSN. It is a Q-learning based routing protocol. It advocates that local information can be communicated and shared with the neighboring nodes with additional overheads. In FROMS, sensor network is represented using a graph G where each vertex corresponds to a sensor node and the edges corresponds to communication channel. Each node which is treated as agent for Q-learning finds the optimum hop cost for any combination of sink. The parameter hop count was basically used during the formation of reward function. In [12], the authors have proposed a Q-learning based energy-aware routing in WSN. By taking into consideration the residual energy of the node, the protocol aims to balance the load among the sensors. The remaining energy of nodes is used in forming the reward and updating the Q-values. The objective is to optimize the network lifetime using load balancing and minimizing the control overhead. Several other authors in [13][14][15] have proposed other reinforcement-based learning protocols that are energy-efficient. The authors in [13] have proposed Q-Routing which takes into account the minimal delivery time to find the best path. Authors in [14] have proposed AdaR that considers residual energy, hop count, aggregated ratio and link reliability to learn and optimal routing strategy, the authors in [15] have proposed QELAR that considers residual energy and energy distribution among the nodes to learn the best path.

VI. PROPOSED ALGORITHM

Since the proposed algorithm in Fig 3 is a variation of original directed diffusion protocol, first of all we will describe the concept of Directed Diffusion (DD) Protocol.

DD consist of interest, gradients, data messages and reinforcements. An interest is the data requirement of the user typically called as query which is flooded to the network by the sink, gradient is the direction from which the interest message was received, data message is the value collected by the sensor as an outcome of some physical phenomenon, reinforcement means that the sink node reinforces or finalizes one or more specific path from among several paths for data communication. DD is a data centric protocol because all the communication is for named data [17]. This Protocol is application aware because design requirements of a sensor network change with application. For instance, an application that focus on delivery time could gather data like nodes' buffer length to get away from the problem of congestion [18][19].

First, we assume that the proposed protocol will have one sink and one source node. Also, each node will have some initial energy and fixed threshold energy. A node will participate in data communication only when its available energy is more than the threshold energy. Whenever the sink node requires some data, it broadcast the data request message to its entire neighbor in the form of a query. The neighbor of the sink continues to broadcast this query to their neighbors until a node detects the required data. If multiple nodes detect the required data, then the node that receives the data first will be considered as the source node. This node is called the source node, S.

```
Sink node broadcast a message to all its neighbors
which in turn continues to broadcast until the source
node sense the target
when source node is detected
do {
    Current node=S
    Available energy of each node= E
    Threshold energy for each node=TE
    path={S}
    For all neighbors $N_i$ of S
    {
        if ($N_i$ is not the destination && E(S)> TE(S))
            {
                Find Q (S, $a_i$)
                E(S)=E(S)-EC(S)
            }
    }
    Determine Max [Q]
    Let Max [Q] corresponds to edge {S, $N_i$}
    Path={S, $N_i$}
    Set current node=Ni;
  }
return path.
```

Fig.3. Proposed Algorithm

Now the algorithm will find an optimum path from the source node to the sink to deliver the requested message. Since the focus of the algorithm is to determine an optimal path from source to sink, we initialize path as π: {S}. The algorithm will add one node to the path in each iteration. For the entire neighbor Ni of Source node S, if Ni is not the sink node and available energy of S is more than the threshold energy, then a Q value for each of the neighbor Ni of S is determined. For finding the Q value, Q-learning algorithm will be used. The neighbor of S that has the maximum Q value will be considered as the next node in the path. If we suppose that Ni has the maximum Q value then the path will be π: {S, Ni}. Now Ni becomes the next node that will be considered as source. In this way the algorithm continues until the Ni becomes the sink node. And finally the algorithm will return the most optimum and energy efficient path π: {S, Ni ,…...,sink}.

VII. Experiments and Results

We have taken a sensor network having 7 nodes which are numbered through 0 to 6. Node 6 is the destination and any other node except 6 can be considered as the source node. Each pair of possible nodes is assigned a distance value. Possible node means that there is a path between the nodes to travel. Node 3 is considered as source in this case. The aim is to find the shortest path from node 3 to node 6 using Q-learning technique so that the reward gained is optimum.

The sensor network has been implemented as a graph (G) using Python. The graph is converted to a matrix (M) whose values are the distance between the nodes. The value zero in the matrix M means that the corresponding pair of nodes does not have a path between them. In Q-learning technique defining a proper reward function is essential to maximize the long-term reward [5]. When the distance between the sensor nodes is more, the energy required to travel will also be large. So, the reward (R) will be inversely proportional to the

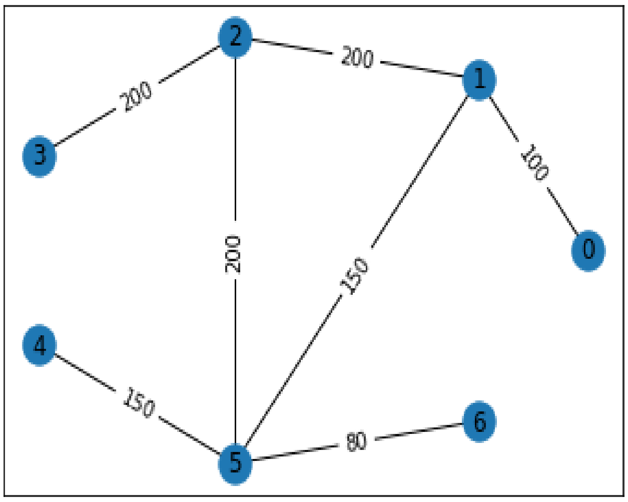

Fig.4. Graph G

$$\begin{vmatrix} 0 & 100 & 0 & 0 & 0 & 0 & 0 \\ 100 & 0 & 200 & 0 & 0 & 150 & 0 \\ 0 & 200 & 0 & 200 & 0 & 200 & 0 \\ 0 & 0 & 200 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 150 & 0 \end{vmatrix}$$

Fig.5. Distance Matrix M

distance (d) between the nodes i.e., R α 1/d. Therefore, in this case, the reward matrix (R) is constructed by taking the inverse values from matrix M. Taking a higher value from reward matrix R implies selecting lower value from distance matrix M and so shortest path will be constructed. The graph G, distance matrix M, reward matrix(R) is shown in Figure 4, Figure 5 and Figure 6 respectively.

In Q-learning, Q-matrix indicates the knowledge that the agent has acquired from experience. In our case each node can be considered as one agent. The row of the matrix Q represents the current node considered by the algorithm and the column represents the possible nodes that can be used to build the path. The agents start out without any previous knowledge and therefore the all values in the Q matrix are initially set to zero.

After the algorithm is complete the Q matrix is modified. Once the matrix Q gets closer enough to the state of convergence, it can be said that the agent learned the best path to the destination. We can construct the path by taking the highest Q value at each state. Figure 7 represents the trained Q matrix produced during our experiment which shows [3, 2, 5, 6] as the most efficient path

$$\begin{vmatrix} 0 & 0.01 & 0 & 0 & 0 & 0 & 0 \\ 0.01 & 0 & 0.005 & 0 & 0 & 0.0067 & 0 \\ 0 & 0.005 & 0 & 0.005 & 0 & 0.005 & 0 \\ 0 & 0 & 0.005 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0067 & 0 \\ 0 & 0.0067 & 0.005 & 0 & .0067 & 0 & 0.0125 \\ 0 & 0 & 0 & 0 & 0 & 0.0125 & 0 \end{vmatrix}$$

Fig.6. Reward matrix R

$$\begin{vmatrix} 0 & 80 & 0 & 0 & 0 & 0 & 0 \\ 80 & 0 & 43.3597 & 0 & 0 & 65 & 0 \\ 0 & 499978 & 0 & 0 & 0 & 55 & 0 \\ 0 & 0 & 43.75 & 0 & 0 & 23.437 & 0 \\ 0 & 0 & 0 & 0 & 0 & 65 & 0 \\ 0 & 59.9999 & 43.7499 & 0 & 56.25 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 100 & 0 \end{vmatrix}$$

Fig.7. Trained Q Matrix

Table 1. Experimental values

	Epsilon=0.9		**Epsilon=0.5**		**Epsilon=0.2**	
Experiment No	**No of Iterations**	**Reward gained**	**No of iterations**	**Reward Gained**	**No of iterations**	**Reward gained**
1	112.098	883.437	107.885	907.30	196.593	871.78
2	119.36	882.038	119.36	884.48	196.593	895.89
3	96.4107	879.704	183.794	870.60	170.113	852.01
4	96.4107	882.038	156.432	909.62	136.13	844.33
5	140.985	882.035	130.393	879.23	186.001	878.90
6	99.9413	912.638	190.855	898.75	142.75	741.76
7	52.2776	881.86	179.381	879.70	181.587	719.65
8	98.6173	893.756	118.036	894.36	191.297	849.34
9	91.9974	886.702	85.8187	919.21	164.817	888.26
10	88.9081	886.702	133.041	904.14	152.901	804.07
Total	**997.006**	**8870.91**	**1404.99**	**8947.4**	**1718.78**	**8346.0**
Average	**49.8503**	**443.545**	**70.2497**	**447.37**	**85.9391**	**417.30**

We have applied the epsilon greedy strategy which adds some randomness when selecting the next node. Instead of always picking the best available nodes, the epsilon greedy approach randomly explores other nodes with a probability epsilon (ϵ) or chooses the best node with probability (1- ϵ). In this way randomness can be added to the algorithm by increasing the value of ϵ. A node that never explores may always be performing sub optimally. Also a node that never exploits may never fully utilize its experience. Therefore, balance between them is very crucial for arriving at optimal solution. The program is run 10 times for different values of epsilon. The result of running the program is shown in Table 1.

It can be seen from the table that when epsilon is set at 0.9, the average number of iterations required for the reward to converge is approximately 49.850305. The reward converges in 70.249785 average iterations when epsilon is at 0.5. And when epsilon is reduced to 0.2 the rewards take on an average 85.9391 iterations to converge. So it can be noted that, algorithms with higher epsilon value converge faster. This is shown by the Figure 9.

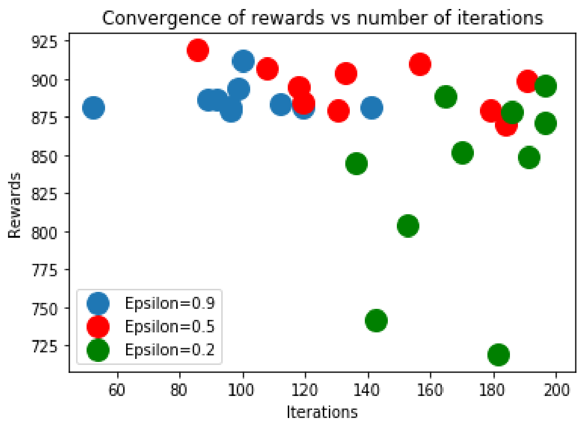

Fig.8. Reward Convergence for different epsilon

We can also observe that the average reward gained is more when epsilon is set to 0.5. When the value of epsilon is set to 0.2 the average reward gained is lowest even when it has taken 85.9391 iterations to reach that value. So, it can be concluded that the balance between exploration and exploitation always yields an optimum value of reward as shown in Table 2

Table 2: Average reward gained at different epsilon

Epsilon	Average Iterations	Average reward gained
0.9	49.850305	443.54552
0.5	70.249785	447.3712
0.2	85.9391	417.3014

VIII. CONCLUSION

Energy Efficiency is an important research area for WSN routing. Reinforcement learning can be used in designing energy efficient routing protocol in WSN. Q-learning is a field of RL that can be applied in WSN to find optimal path. In this paper RL-based routing algorithm was proposed that is applicable in WSN lifetime optimization problem. The algorithm chooses the next forwarding node using previously learned data and the currently estimated data. Parameters such as distance between the sensor node, number of hops, remaining energy of the node etc. can be considered in defining the reward function in Q-learning algorithm. Such energy efficient routing protocol can enhance the overall performance of WSN. Also the significance of balancing the exploration and exploitation in Q learning is demonstrated using Python. The result produced partially verifies that if exploration and exploitation is properly balanced then the Q-learning algorithm always produces optimum rewards. Therefore, if the reward function is properly defined considering the requirement of the application, the proposed algorithm is expected to give an improved performance than the original Directed Diffusion Protocol.

REFERENCES

[1] Akyildiz I F, Su W, Sankarasubramaniam Y, Cayirci E.(2002), "A survey on sensor networks", IEEE Communications Magazine 40 (8) , pp. 104–112.

[2] Rault T, Bouabdallah A, Challal Y(2014), ''Energy efficiency in wireless sensor networks: A top-down survey,'' Comput. Netw., vol. 67, pp. 104–122. Jul 2014.

[3] Yadav S, Yadav R S (2015), "A review on energy efficient protocols in wireless sensor networks". Wireless Networks, 22(1), pp. 335-350 2015.

[4] Halawani S, Khan A(2010), "Sensors Lifetime Enhancement Techniques in Wireless Sensor Networks - A Survey", Journal of Computing, vol. 2, issue 5, pp. 34-47,May 2010.

[5] Guo W, Yan C, Lu T(2019), "Optimizing the lifetime of wireless sensor networks via reinforcement-learning-based routing," International Journal of Distributed Sensor Networks, vol. 15, no. 2, p. 1550147719833541, 2019

[6] Sutton .R S, Barto A G(2014), "Reinforcement Learning: An Introduction", II ed, The TMT Press, Cambridge, Massachusetts,2014.

[7] Patel A B, Shah H B(2015), "Reinforcement Learning Framework for Energy Efficient Wireless Sensor Networks", IRJET, Volume 2, Issue 2, pp. 1034-1040, May 2015

[8] Watkins C J C H, Dayan (1992), "Q-learning". Machine Learning, 3:279–292, 1992

[9] Amjad N, Sandhu M M, Ahmed S H.et al(2013), "DREEM-ME: distributed regional energy efficient multi-hop routing protocol based on maximum energy with mobile sink in WSNs," Journal of Basic and Applied Scientific Research, vol. 4, no. 1, pp. 289–306, 2013.

[10] Renold A P , Chandrakala S(1996), "MRL-SCSO: multi-agent reinforcement learning-based self-configuration and selfoptimization protocol for unattended wireless sensor networks". Wirel Pers Commun 2017; 96: 5061–5079.

[11] Förster A, Murphy A L(2007), "FROMS: Feedback routing for optimizing multiple sinks in WSN with reinforcement learning," in Proceedings 3rd International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2007.

[12] Oddi G, Pietrabissa A, Liberati F(2014) , "Energy balancing in multi-hop Wireless Sensor Networks: an approach based on reinforcement learning", 2014 NASA/ESA Conference on Adaptive Hardware and Systems (AHS), 2014

[13] Boyan J A, Littman M L(1993) "Packet Routing in dynamically changing network: a reinforcement – learning approach". In: Proceedings of the international conference on neural information processing system, Denver, CO,,29 November-2 December 1993, pp.671-678. New York:IEEE.

[14] Wang P , Wang T(2006) , "Adaptive routing for sensor networks using reinforcement learning". In: Proceedings of the IEEE international conference on computer and information technology, Seoul, South Korea, @0-22 September 2006, pp.219-224. New York: IEEE

[15] Hu T, Fei Y(2010), " QELAR: a machine learning-based adaptive routing protocol fro Energy efficient and lifetime-extended underwater sensor networks". IEEE T mobile Comput 2010; 9(6): pp.796-809.

[16] McCullock J, "*A* painless Q-learning Tutorial", Accessed on Feb 02 2020 [Online], Available on:https://mnemstudio.org/path-finding-q-learning-tutorial.html.

[17] Intanagonwiwat C, Govindan R., Estrin D, Heideman J, Silva F(2002), "Directed diffusion for wireless sensor networking," IEEE/ACM Trans. Networking, vol. 11, pp. 2–16, Feb. 2002.

[18] Samara K, Hosseini H(2016), "Aware Diffussion: A semi-holistic routing protocol for Wireless Sensor Network", Wireless Sensor Network, Vol 8, pp.37-49,2016.

[19] Liu J, Li Y, Chen Q, Kuang Y, et al.(2007), "Energy andStorage Efficient Directed-Diffusion for WirelessSensor Networks:. In Proc. 2007 InternationalConference on Wireless Commuications, Networkingand Mobile Computing, Proceedings-Volume 4.Sep.21-25,2007.

AUTHOR PROFILE

Tirtharaj Sapkota is currently working as a faculty member in the Department of Computer applications, Purbanchal University, Nepal for last 16 years. He has completed Master of Computer Application from Dibrugarh University, India in the year 2003.He is currently a PhD research scholar at Assam Don Bosco University, Guwahati, India. His areas of interest in research are Machine Learning, Wireless Sensor Network and IoT.

Dr. Bobby Sharma is currently associated with Assam Don Bosco University as Associate Professor and Head of Computer science and engineering Department. She has more than 18 years of teaching and research experience and about 6 years of Industrial Experience. Her areas of interest include Network security, Wireless Sensor Network, IoT, IoE, and Artificial Intelligence.