Q Learning Based Adaptive Protocol Parameters for WSNs

Piumika N. Karunanayake, Andreas Könsgen, Thushara Weerawardane, and Anna Förster

***Abstract*—Wireless sensor networks (WSN) are widely used for multi-disciplinary applications. According to the requirements and the goal of the application, the network is designed and the protocol is tuned to obtain the best performance of the WSN. In real world applications, all nodes in the network have a common protocol parameter set, irrespective of their position in the network. In several experiments with multihop sensor networks, we observed that individual nodes perform differently depending on the protocol parameter values. With the observation the question was raised whether the performance of the network can be improved by using tuned parameter sets for each individual node in the network. Tuning protocol parameters for each node manually is tedious and may not be practical for large number of nodes. As a solution, adaptive protocol parameters are introduced using reinforcement learning. The learning algorithm gradually approaches an optimal set of protocol parameter values for each and every node during the runtime resulting in average improved network performance with 13.44% and 29.41% compared to networks with static common parameter sets in a network of 20 and 30 nodes respectively in simulation environment. The performance of the adaptive protocol is validated using real testbed with 10 nodes and the performance improvement is 16.21%. With the simulation results it was observed that networks with higher number of nodes obtain more performance gain using the adaptive protocol algorithm compared to networks with lower of nodes.**

***Index Terms*—Adaptive protocol parameters, Q learning, WSN.**

I. INTRODUCTION

WIRELESS sensor networks (WSN) emerge as an effective technique to gather information from a potentially large amount of sensor nodes. The applications of WSNs vary from industrial, environmental, medical, habitat monitoring to domestic applications [1]–[3]. Each application constitutes its own goals and limitations. To achieve the goals while tackling all possible constraints, WSN protocols need to be fine-tuned according to the application. There are several protocol parameters which need to be optimized according to the application, topology and number of nodes in the network. Applying the same WSN protocol or protocol parameters for different applications will reduce the reliability, time efficiency or the energy efficiency of a WSN. Also it leads to undesirable performance of the network. Hence it is mandatory to adjust the protocol parameters according to the application which is expedited by experts based on experience, analysis of expected traffic load or by performing a number of trials before actual deployment. Therefore the WSN designer undergoes a tedious process to identify the optimal parameter set before deploying the network even common set of parameters are activated for all the nodes in the network. Hence to obtain the optimum parameter set for each individual node, machine learning technique is introduced which learns from the environment to identify the best parameter set for each node.

The sensor nodes play different roles within multi-hop networks compared to single-hop networks. A set of nodes acts as leaf sensor nodes and remaining nodes relay the packets to the sink including its own packets. Each node handles different traffic load according to the placement of the node in the network. Also the packet collisions and the interference the nodes experience depends on the placement of the node. In this paper we show that optimizing parameter values for individual node improves the performance of the network.

By analyzing the current status of the network, a parameter reconfiguration approach can be specified either in a decentralized or centralized manner. In the centralized manner all the information is passed to a central node which decides whether a reconfiguration needs to be executed and if so which nodes should be reconfigured. This method results in large overhead due to the amount of exchanged information. In centralized networks, frequent changes of the network is not catered due to the mentioned overhead. In decentralized methods, nodes take decisions locally by observing its own quality measurements.

There are two methods that can be used for parameter optimization, mathematical optimization and machine learning [4]. In case of linear objectives and constraints, linear programming can be used for convex optimization problems, or an equivalent Lagrange dual problem can be constructed in mathematical optimization. A Lagrange dual problem is solved in a distributed manner. Therefore it can be applied for WSN as well [4]. To use mathematical optimization in a network an extensive knowledge about the network is required. Hence it is mandatory to have prior knowledge about the network and considerable amount of information passing within the network on top of normal traffic load. A survey was conducted to evaluate the pros and cons of using machine learning and fuzzy logic as an adaptation mechanism in WSNs [5]. In [6]

Manuscript received March 17, 2022; revised July 16, 2022; approved for publication by Jiming Chen, Division 2 Editor, August 13, 2022.

P. N. Karunanayake is with Department of Sustainable Communication Networks, University of Bremen, Germany and with Department of Electrical Electronics & Telecommunication Engineering, General Sir John Kotelawala Defence University, Sri Lanka, email: piumikan@comnets.uni-bremen.de.

A. Könsgen and A. Förster are with Department of Sustainable Communication Networks, University of Bremen, Germany, email: {ajk, anna.foerster}@comnets.uni-bremen.de.

T. Weerawardane is with Department of Electrical Electronics & Telecommunication Engineering, General Sir John Kotelawala Defence University, Sri Lanka, email: tlw@kdu.ac.lk.

P. N. Karunanayake is the corresponding author.

Digital Object Identifier: 10.23919/JCN.2022.000035

and [7] the applicability of machine learning techniques to improve WSN performance have been studied extensively. Compared to mathematical optimization, machine learning requires a less amount of information regarding the network.

Optimizing the parameters can be carried out in two situations, static environment and dynamic environment. The objective of optimizing the parameters in static environment is to obtain best parameter set for a particular application when using a standard or commonly used protocol. Obtaining optimum parameters in dynamic environment focuses on maintaining the performance by adapting the parameters according to the dynamics.

In this paper a novel algorithm for adaptive parameter adjustment is proposed for multi-hop networks which operates without prior knowledge of the network or the channel in a static environment. The adaptive parameter algorithm provides optimum parameters for each individual node for any application without any modification of the original protocol. The most impacting protocol parameters which need to be optimized for the application are decided by the network designer. For the algorithm, the prior knowledge of the impact to the network when parameter values are changed is not required. The network performance after parameter values are changed is learned by the algorithm which is based on Q learning, during run time in a distributive manner using a reactive approach. This results in obtaining optimized parameters for each node without injecting additional signalling packets as they would be needed in a centralized approach. Since each node seeks for its best parameter set with the adaptive protocol parameter algorithm, the network consists of nodes which operates with different parameter values.

Using simulations and experiments with real nodes, we show that the proposed adaptive algorithm outperforms the configuration of the protocol with static parameter values.

The rest of the paper is structured as follows. We demonstrate our algorithm using the well known Collection Tree Protocol (CTP) [8]. Therefore the background of CTP is presented under Section II to provide a clear view of our usage of the protocol for simulations and real world applications. Section III discusses previous work related to adaptive parameters for WSN and application of reinforcement learning algorithm in WSNs. The motivation for this work is discussed in Section IV. The model of the adaptive protocol parameter algorithm is discussed under Section V whereas Section VI discusses the design of the algorithm. The simulation setups and the real world experiments setup which were used to validate the usage of adaptive protocol parameters using reinforcement learning is presented in Section VII. Section VIII covers the results obtained and discusses the findings. Section IX, includes the conclusion remarks.

II. Collection Tree Protocol

To evaluate the performance of the proposed adaptive parameter algorithm, the Contiki implementation of CTP is used due to its deployment in practical applications. Although CTP is selected for evaluation of the algorithm, any other protocol may be used with the algorithm. This section provides a brief description of CTP. Subsection VI-A discusses how parameters are selected in general for optimization and how it is applied to CTP protocol.

CTP is a tree based collection protocol. All the nodes send packets towards the sink node in multi hop network by considering the expected number of transmissions (ETX) value of its own and the neighbouring nodes. ETX value is the expected number of packet transmission for one packet to reach the sink node, it is used to measure the cost of a link. CTP mainly focuses on achieving reliability, robustness, efficiency and hardware independence [8]. The protocol has proven that all the mentioned goals are achieved with a series of experiments in real-world testbeds. The protocol achieves its goals by improving the accuracy of the link estimation, by continuously validating the data path and using adaptive beaconing. Each method is described in the following paragraphs.

Improving the accuracy of the link estimation is carried out by combining information from the physical layer, data link layer and routing layer to calculate the link estimate. This overcomes the challenges which rise due to intermediate-quality links, time-varying nature of wireless channels, multi-path inter-symbol interference and hardware variations. The physical layer provides the quality of the received packets while the data link layer gives the information on the ETX value of each link. The network layer improves neighbour discovery by activating route comparison.

All nodes keep track of the cost to the sink node. Once a packet is received by the node it checks the cost of the sending node. If the cost of the sending node is lower than the receiving node, the received node detects an inconsistency in the cost update which will end up with a loop in the network. If inconsistency is detected, topology adaptation is triggered and using the Trickle algorithm [8] the new information is passed throughout the network. By checking all the received packets, the nodes maintain the data path validation.

The third method uses the adaptive beaconing to maintain the consistence cost values of the links. Once the beaconing period is low, the cost information will be up-to-date whereas the overhead will be high. Also it consumes high bandwidth and high energy. Large beaconing periods would result in low bandwidth requirement and low energy consumption but would result in inconsistent cost value updates of links leading to loops within the network. The beaconing period shrinks due to three events:

- Detect inconsistency in ETX values.
- Detect superior path.
- Receive a request for a beacon.

The beaconing period is expanded once the node hears another node sending the packets with information which node already has.

In CTP, transmit timers are also adjusted to minimize the self-interference. If the expected packet time is p, a node always waits for a duration of 1.5 to 2.5 p to send a packet. Hence once the parent node forwards the packet, it will not collide with the child's second packet.

There exist a number of parameters in the implementation of CTP. These parameters focus on efficiently calculating the link cost, packet transmission and receiving, maintaining the packet queue and beaconing. Out of these parameters three parameters were selected to incorporate the proposed algorithm. Subsection VI-A discusses the selected parameters for optimizing and their attributes used in the adaptive learning algorithm.

III. Related Work

Wireless sensor networks are self maintained devices in most applications. Therefore in recent years adaptation has been incorporated to different protocol layers in different designs of wireless sensor networks.

In [9] it is shown that the MAC layer protocol needs to be selected according to the application scenario and the condition of the environment. Hence the authors have proposed to select the optimum MAC layer protocol according to the frequently varying condition of the underwater WSN using a software defined communication stack (SDCS). In our work a single protocol functions with optimum parameters without changing main objectives of the protocol.

The authors of [10] have showed that by using adaptive low power listening, the energy consumption of the WSNs in noisy environment can be improved. The problem is solved considering it as an optimization problem. Therefore to minimize the computational burden for individual sensor node, optimal values are pre computed and each nodes keeps the values locally. This requires knowledge about the network and the environment whereas in our work pre knowledge is not required. To maximize the network reliability, life time and to reach the constraints of end to end delay, an adaptive mechanism is introduced for the wake-up rate and the transmit radio power of each node [11]. Again this work illustrates how the performance improvement is gained by considering it as an optimization problem. Adaptive duty cycling is introduced in [12] to handle load variations in time and location, which leads to lower energy consumption in WSNs. The Previously mentioned work require an awareness of the network and the application, however in our work, the machine learner collects information about the network during run time.

For the protocol IEEE 802.15.4, with two assumptions, a single hop network and infinite queue length, authors of [13] have introduced tuning parameters for the CSMA/CA mechanism in a distributed manner. The algorithm performs well for static networks as well as for dynamic networks. An adaptive backoff algorithm is introduced in the work of [14] which improves the throughput and reduces the energy consumption in the network. The main assumptions are not required in our approach which performs well with multihop networks and limited queue length.

The authors of [15] and [16] have used a proactive distributed approach to obtain adaptive parameters for the network dynamics. In this method, all the dynamics should be identified and parameter values should be defined accordingly which is not a requirement in our work .

In order to maintain the minimum required QoS, [17] proposes a solution using a distributed feedback control mechanism. End to end delay and delivery ratio have been optimized using two controllable parameters, the transmitting power and the number of retransmissions.

To maximize the battery life time, the authors of [18] have proposed protocol optimization using the status of the battery power for IEEE 802.15.4. The authors of [19], [20] have proposed an algorithm to obtain optimum parameter settings for 802.15.4 WSNs. The proposed algorithm of [19] adopts optimum parameter values when the operating conditions change over time. All of the above work optimizes a particular protocol but our proposed method is possible to work on top of any protocol without constraints.

Reinforcement learning is used in many WSN related works due its simplicity in the implementation, low complexity and its capability of learning from the environment without prior knowledge. In [21], the sampling interval is adjusted to minimize the number of transmissions according to the changes in the environment. Also the work highlights the importance of the proper choice of learning parameters for its performance. In [22] reinforcement learning is applied in underwater WSN to improve the packet delivery ratio and the energy consumption.

The authors of [23], [24] have used reinforcement learning for energy management maintaining network performance without an energy consumption tracker. The QL-Mac protocol is introduced in [25] which is an energy efficient protocol utilizing Q learning to adapt the duty cycle. Q learning is incorporated to the ALOHA protocol in the work of [26] resulting in an efficient transmission schedule. The introduced algorithm is robust in the dynamic environment and during packet losses due hardware issues. An efficient routing algorithm is introduced in [27] incorporating Q learning as the learning algorithm. In [28] the communication range is adjusted using reinforcement learning to control the number of connectivities per node resulting in a reduction of energy wastage. The authors of [29] have proved by using reinforcement learning that the transmit power can be adjusted considering the feedback of the data packets, hence extending the life time of the network. A secure routing scheme is implemented to avoid malicious nodes in the work of [30] using reinforcement learning.

The research work mentioned in the last two paragraphs leverage reinforcement learning to improve the performance of the network and most of them focus on particular application or one specific parameter. Our work can be distinguished from the previous work by introducing an algorithm which can be incorporated without changing the properties of the underlying protocol. Also the algorithm works independently from the network topology without additional burden to the network.

IV. Motivation: Evaluation of Node's Individual Performance with Different Protocol Parameters in a Multi Hop Network

To identify the impact of the protocol parameters on the performance of individual nodes, a series of experiments were carried out with different static protocol parameter sets in multi-hop networks.

TABLE I
PARAMETER SETS FOR PRACTICAL EXPERIMENTS. PACKET RATE: ANIMAL DETECTION: 1 PKT/MIN; TEMPERATURE SENSING: 1 PKT / 12 MIN, JAMMED.

Parameter	Set 1	Set 2	Set 3
MAX_MAC_REXMITS	2	4	2
MAX_ACK_MAC_REXMITS	2	4	2
REXMIT_TIME	4sec	4sec	2sec

Two different application scenarios were selected, the first application is temperature sensing which requires a low data rate in a highly interfered area assuming an urban or indoor environment. The packet rate was considered as one packet per 12 minutes. The second application is animal detection which requires a high data rate in a low interfered area assuming large field with crops where interference is low. For this application the packet rate was considered as one packet per 1 minute. Both application scenarios were experimented in a real network with 3 different sets of protocol parameters. The first parameter set includes default parameter values. In the second parameter set, the max. number of retransmissions was doubled w. r. t. the default settings and in the third set, the retransmission time was halved. Rime is a lightweight layered communication protocol stack for sensor networks and implemented using Contiki OS [31]. The protocol stack consists of the Radio layer, RDC layer, MAC layer and Network layer.[1] For the experiment, CTP was used on top of the Rime protocol, ContikiMac in the RDC layer and the CC2420 chip as the hardware interface in the radio layer. The experiments were carried out with 9 Zolertia Z1 nodes creating a multi-hop network. To generate interference, another Z1 node was placed as a jamming node for the application of temperature sensing. For each parameter set, an experiment running 24 hours was conducted. The results are compared between the two applications for the same topology. Table I illustrates the 3 sets of local parameters.

Table II illustrates the performance of each node measured by the ETX value.

Nodes 7, 8 and 9 have better ETX value with the parameter set 1 for the animal detection scenario whereas nodes 2,3 and 6 perform well with parameter set 2 for the same application scenario. For Temperature sensing application, all nodes perform well with the parameter set 2. The evaluation of the results highlights that nodes perform in a distinct manner for different parameter sets, although the application scenario is same. Considering the two application scenarios, the best parameter set for a node is not identical to the best parameter set of the same node in another application. The main reason for this is the traffic load handled by each node and the varying demand for channel access among the nodes. Therefore we propose an algorithm to identify a better parameter set for each node in a distributed manner maximizing the performance of the network compared to the performance of a network with original static protocol parameters. The performance of the network is measured by the packet delivery ratio.

[1]http://anrg.usc.edu/contiki/index.php/Network_Stack

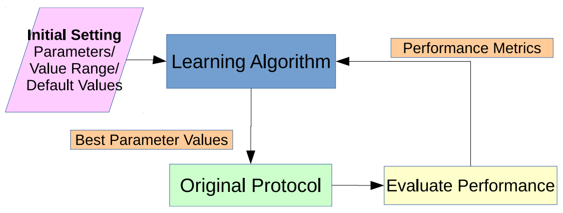


Fig. 1. Extension of the original CTP by Q learning.

V. MODEL OF THE ADAPTIVE PROTOCOL PARAMETER ALGORITHM

The goal of our work is to increase the performance of the network by optimizing protocol parameters for individual nodes without additional effort for the network designer. Examining the parameters in a centralized manner and evaluating the best parameter set, followed by sending the optimum values for the individual nodes will be a huge burden for the network. Therefore in the proposed work, individual node evaluates its own performance and converges to a set of parameters that maximize the performance. The significance of the algorithm is that for any topology or any number of nodes it should identify a better set of protocol parameters compared to the original static parameter set.

The model of our proposed algorithm is illustrated in Fig. 1. It consists of three main parts, the input, the machine learner and the output.

Three types of inputs are considered: The parameters which need to be optimized, the range of each parameter and the default value for each parameter.

To select the protocol parameters, an analysis of parameters was carried out in general. There are two types of protocol parameters, local parameters and global parameters. The local parameters affect the functionality of individual nodes with minimum impact on the other nodes in the network. For example, the maximum number of retransmissions, maximum number of ACK retransmissions or length of the sending queue. On the other hand, global parameters have a direct impact on the functions of all nodes, for an example the duty cycle. Since our work focuses on optimizing parameters individually, only local parameters are considered. In the case of CTP, *maximum number of MAC retransmissions, backoff time for retransmission, minimum available queue entries for own packets* were considered for optimization and provided as the user input to the model. Although we selected three parameters to be improved by the algorithm, the number of parameters to be optimized can be decided by the network designer.

The selected parameters for optimization will have a restriction w. r. t. the value range from the original protocol implementation. Therefore for each parameter, the value range is required as the input for optimization as the second type of input. The third input is the best known value of each selected parameter. By starting from the default or best known value, the algorithm converges to a parameter combination which provides better performance compared to the original protocol which uses the default value. As for the experiment purposes, the Contiki implementation of the CTP protocol was selected

TABLE II
ETX VALUES OF ANIMAL DETECTION AND TEMPERATURE SENSING APPLICATION.

Sensor node	ETX value for animal detection			ETX value for Temperature Sensing		
	Set 1	**Set 2**	**Set 3**	**Set 1**	**Set 2**	**Set 3**
Node 2	3.26	1.81	2.15	3.19	1	2.16
Node 3	2.39	2.32	3.22	3.23	2.1	2
Node 4	2.12	2.44	4.36	5.32	3	2.16
Node 5	1	1.01	1.02	2.02	1	1
Node 6	2.64	2.01	3.33	2.18	2	2.04
Node 7	2.55	3.05	3.33	3.25	2	3.56
Node 8	1.65	2.73	1.04	1.04	1	2
Node 9	3.13	3.8	2.09	3.04	2	3.8
Mean	2.34	2.39	2.57	2.9	1.76	2.08

due to practical reasons. The parameter selection for adaptation in CTP is discussed in Subsection VI-A.

The machine learner evaluates the performance of different combinations of protocol parameters to output the best combination. There are many approaches for the learner to compute the best combination of parameters as offline computation, model-based adaptation and measurement-based adaptation [32]. As processing is carried out within each individual node, the implementation is required to be simple utilizing a minimum amount of memory and processing. Q learning includes the mentioned properties and in addition, it is model-free and adapts to changing environments and falls under measurement based adaptation. Therefore, as the learning algorithm to identify the optimum set of protocol parameters, Q learning is selected. The learning algorithm implemented in the adaptive parameter protocol, is discussed in more detail in Subsection VI-B.

The output of the model will be, an optimized parameter value set for each individual node, which leads to improved overall performance in the network exceeding the performance of the original protocol.

VI. Adaptive Protocol Parameter Algorithm with Q Learning

The proposed algorithm consists of two main components. Selection of parameters and its attributes is the first component whereas learning and optimizing the parameters is the second. The two components are discussed under two Subsections VI-A and VI-B.

A. Selection of Parameters and Their Attributes

A protocol consists of a number of parameters which affect the performance of the network. Tuning or adapting all parameters is not practically feasible. Therefore for adaptation it is important to identify parameter set which would highly impact on the performance of the network. Once the parameter set is identified, the value range for each parameter needed to be determined. To overcome the difficulties in identifying the best value range, the default value of each of the parameters is considered and a value range is selected which includes adjacent values of the default value. For the implementation, the minimum value, maximum value and the unit step size is provided as the input.

TABLE III
PARAMETERS AND THEIR ATTRIBUTES.

Parameter	Default values	Value range
Max number of MAC REXMITS	2	[1, 5]
Minimum allocated queue entries	2	[1, 5]
Retransmit backoff (sec)	4	[1, 5]

Let X be a selected adaptive parameter. It consists of a value set that can be assigned as shown in (1).

$$X = x_1, x_2, \cdots, x_n \tag{1}$$

Let x_1 be the minimum value and x_n be the maximum value, the default value is within this set. The unit step value is the difference between two adjacent values of the set. If the unit step is 1, then $j = i + 1$ where x_i and x_j are adjacent values in the set. The difference between any adjacent values is considered to be constant. In the adaptive algorithm, if the exploration is conducted, the algorithm changes the parameter value either by increasing or decreasing it in unit step. Therefore the parameter value cannot jump arbitrarily to any value within the set. The reason behind the constraint is that the assumption of performance improvement or reduction changes gradually along the parameter set.

If there are k adaptive parameters and n is the number of values for each parameter, the optimum set of parameters needs to be determined out of n^k combinations or states. During the exploration, one parameter set which includes adjacent parameter values with the existing set out of six possibilities will be selected using the random generator.

For optimization purposes three parameters were selected from the CTP protocol: maximum number of retransmissions, minimum allocated queue entries for own packets, and retransmission backoff time. These parameters were selected as they are local parameters and considerably contribute on the packet delivery rate. The default values assigned for each parameter were 2, 2, and 4, respectively. Hence for all three parameters, the value range was decided as [1,5]. Therefore as the inputs to the algorithm were given as mentioned in Table III.

B. Adaptive Algorithm Based on Q Learning

The system requires learning from the environment without prior knowledge. Hence we prefer reinforcement learning over supervised learning and unsupervised learning algorithms. Q learning and SARSA fall under reinforcement learning

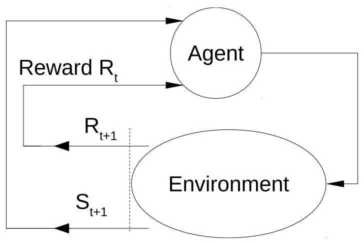


Fig. 2. Agent-environment interaction in reinforcement learning adopted from [33].

theories, but Q learning is selected over SARSA as it converges faster compared to SARSA. Also Q learning is simple, model-free, adaptable to changing environments and uses less processing power with lower amount of memory [33]. Also it allows to evaluate a better parameter set in a distributed manner.

An agent in Q learning is a learner and a decision maker who takes actions according to a policy and interaction with the environment. For the actions that are taken, rewards are allocated according to the reward function. An action value is denoted as $Q(t, a)$ which implies the estimated value on the t^{th} time step [33]. Fig. 2 illustrates the link between the agent and the environment.

The main Q learning components related to the our adaptive algorithm are shortly described below.

- State: Valid protocol parameter combination which requires adaptation.
- Action: Increasing, decreasing or keeping the same value for the parameter.
- Reward: The reward received after evaluating the performance metrics, number of queue drops and expected number of transmissions.
- Exploration: Generate random value and if the random value is less than ϵ, a random action is generated out of six possible actions, which is increasing or decreasing one parameter at a time. Keeping the same value for all three parameters state was not considered for exploration.

The ϵ-greedy method is one method used for action selection. In this method, exploration is conducted with a probability of ϵ. The ϵ value can be reduced over time. The disadvantage of the ϵ-greedy method is, when the agent wants to explore it considers all actions with an equal probability. For this reason the worst action or actions are selected with a probability of other desirable actions. To overcome the problem of selecting any action with equal probability during exploration, the softmax action selection rule is proposed in the literature [33]. The disadvantage of the softmax strategy compared to the ϵ-greedy method is the difficulty in tuning parameters of the softmax strategy [34]. For the implementation of our algorithm, ϵ-greedy is used for its on-node implementation efficiency [35].

Once values for the parameter set are selected according to the ϵ-greedy policy, values are not changed until the node transmits three self generated packets. A window size of three packets for reward calculation was decided based on the performance of the network after conducting series of simulations. The reward is calculated according to following algorithm:

Algorithm 1 Reward assignment

```
1: procedure CalRwd (cTx,prevTx,cDrop,minDrop,hop)
2:     $i \leftarrow 0$
3:     if $cTx = hop$ && $cDrop = 0$  then $i \leftarrow 4$
4:     else if $cDrop = 0$  then $i \leftarrow 3$
5:     else if $cTx \leq prevTx$ && $cDrop \leq minDrop$  then $i \leftarrow 2$
6:     else if $cDrop \leq minDrop$ then $i \leftarrow 1$
7:     else $i \leftarrow 0$
8: return $i$
```

For the reward calculation, two performance measurements of individual nodes are considered. The number of transmissions required to send a packet to the sink and number of drops in the send queue. With the two measurements, the algorithm monitors the queue drops and the number of transmissions required to send one packet and provides reward for the parameter combination. After transmission of 3 packets, the performance is evaluated for the window. The algorithm for reward calculation is illustrated under Algorithm 1.

As we start from the known best values which may not be optimum, positive values are allocated for rewards. A node's best performance occurs in a situation where the number of transmissions from the nodes to sink is equal to the number of minimum hops and zero packet drops. If a node achieves the mentioned criteria, maximum award is assigned. For all other cases which are better than the best known situation, positive reward is allocated and if there is no improvement, zero reward is allocated.

Once the reward is given for the performance of the parameter set within the window, the Q value is updated. For the updating rule user defined two parameters are used, step size parameter α and discount parameter γ. The step size parameter confirms that the Q value is the weighted average of past rewards and the initial reward estimate. The step size parameter should be selected confirming the convergence of the Q value with time. For the convergence, two conditions needs to be satisfied as illustrated in (2) and (3) by the value assigned for the step size [33]. $\alpha_k(a)$ is the step size parameter when action a is selected for the k^{th} time step.

$$\sum_{k=1}^{\infty} \alpha_k^2(a) < \infty \tag{2}$$

$$\sum_{k=1}^{\infty} \alpha_k(a) \to \infty \tag{3}$$

The discount factor γ determines the impact of the future expected reward for the current reward. The value of the γ varies in the range of [0,1]. When higher discount factor is selected, future rewards are taken into consideration where for lower discount factor more focuses on the current value.

Initial values in the parameter set are known best values. The effort of the algorithm is to examine whether there is a better set than the known parameter set. For this reason the algorithm

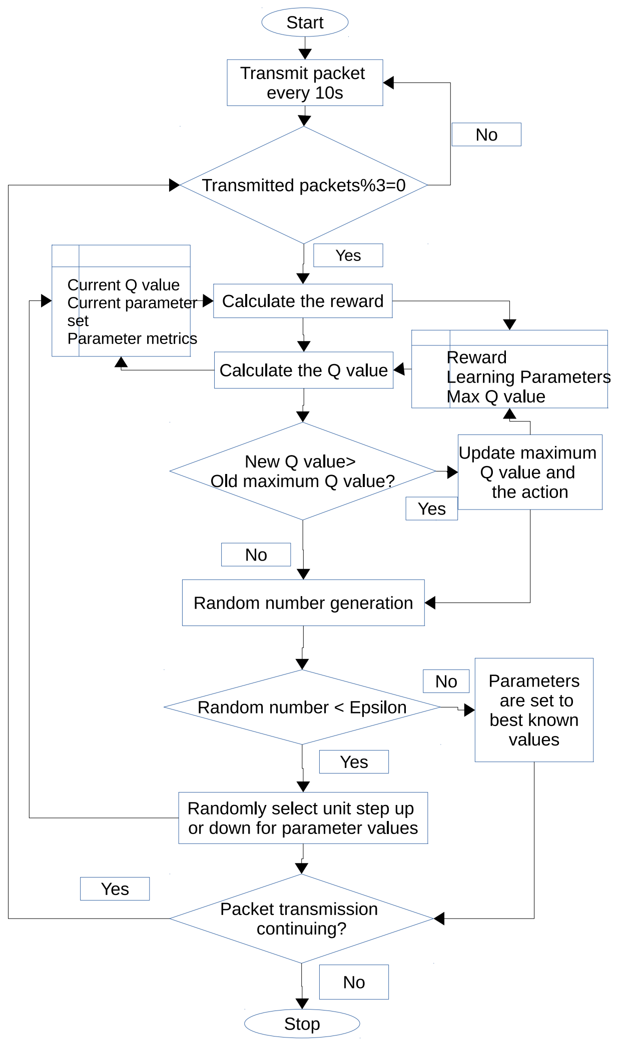


Fig. 3. Flow chart of the adaptive learning algorithm.

needs to explore more before selecting another parameter set as the optimal solution. Also the optimal solution must be selected with considerable number of observations. For this reason, the Q value that has the action converging towards the known best parameter combination is set to 1, which is a non-zero positive value in order to encourage exploration before convergence. The rest of the Q values are set to zero. All the Q values are stored in a static array and the values are updated according to the update rule each time exploration or exploitation is carried out.

The update rule for Q values is as follows:

$$Q(S,A) \leftarrow Q(S,A) + \alpha \cdot (R + \gamma \cdot \max_a Q(S',a) - Q(S,A)), \tag{4}$$

where $Q(S,A)$ is the Q value for particular state and action, α is the step size, R is the reward obtain for taking the action A in the state S and γ is the discount factor. $Q(S',a)$ refers to the Q value of the new state taking the action a. With the update rule, new Q value for that parameter set with the action is calculated by considering the reward for taking that action and maximum expected future reward given in the new state.

Since our work is focused on a static environment, initially learning parameters were set with the best combination and ϵ was reduced after the network reaches the stable state. Hence initially the ϵ value was set to 0.1 and after 300 exploration or exploitation ϵ value was reduced as the packet delivery rate (PDR) value tend to converge. In each learning cycle after 300 explorations, the ϵ value is multiplied by a constant which is less than 1. A cycle includes a number of self-generated packets specified by the window size w which is set to 3. Once a node transmits w packets, ϵ is updated as shown in (5).

$$\epsilon_{t+w} = \epsilon_t \times 0.99993, \tag{5}$$

where ϵ_t is the current ϵ value and ϵ_{t+w} is the new value after the number of packets specified by w have been transmitted. Therefore once the convergence is initiated, the probability of exploration is reduced.

The exploitation or exploration is also initiated at the beginning of a window of 3 transmitted packets. The window size was decided after conducting a series of simulation experiments with a network of 10 randomly deployed nodes. If the packet window is long, it will give better performance evaluation but the convergence gets delayed. If the packet window is too short, accurate performance of the parameters cannot be obtained hence reward and the Q value calculation will not be accurate.

The flow chart of the algorithm is illustrated in Fig. 3.

VII. Evaluation Methodology: Simulation and Real-World Experiments

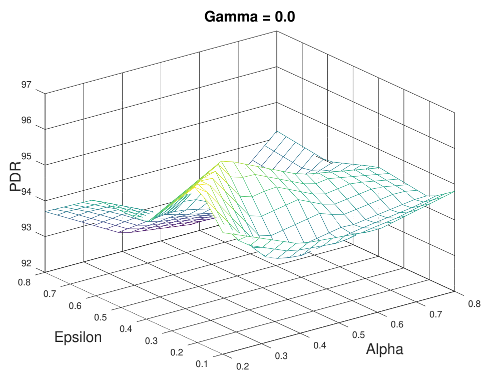
A. Methodology for Simulation

To evaluate the performance of the adaptive protocol, the Cooja simulator[2] was used. Cooja was selected as it has already implemented CTP using Contiki and for real world testing, nodes could be programmed directly with the code developed in Cooja [36], [37]. Z1 sensor nodes were selected as the node model and the unit graph disk medium (UGDM) distance loss model was used to simulate the wireless medium.

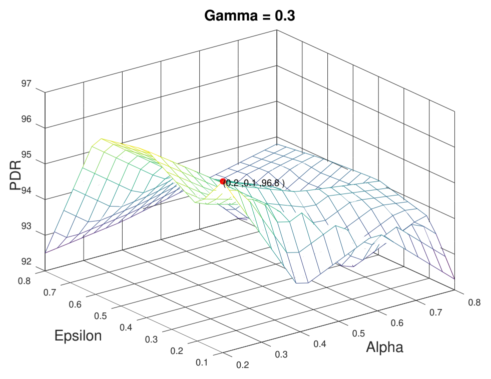
The simulations were conducted for networks with 10, 20 and 30 nodes. Each set of nodes were randomly deployed for 12 different topologies and each topology was repeated with different random seeds for 10 times. Networks with 10 nodes were deployed within an area of $125\,\mathrm{m} \times 125\,\mathrm{m}$, in case of networks with 20 or 30 nodes the area was $150\,\mathrm{m} \times 150\,\mathrm{m}$. For all topologies, the average packet transmission rate was one packet per 10 seconds, the time interval between two consecutive packets were randomized to avoid collision. The Rime protocol was used on top of ContikiMac as the MAC protocol. The simulations were terminated once each node had sent 2500 packets.

Initially each adaptive parameter was set to the default value as in the original protocol and the parameter range was defined including five adjacent values. The minimum, maximum and unit step values were given as the input for the adaptive algorithm.

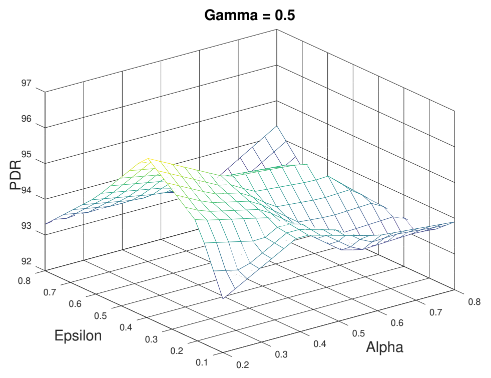
[2]http://anrg.usc.edu/contiki/index.php/Cooja_Simulator



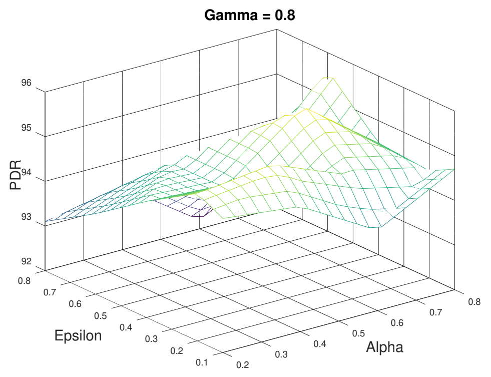
(a)



(b)



(c)



(d)

Fig. 4. PDR for different fixed settings of the parameters α, ϵ and γ.

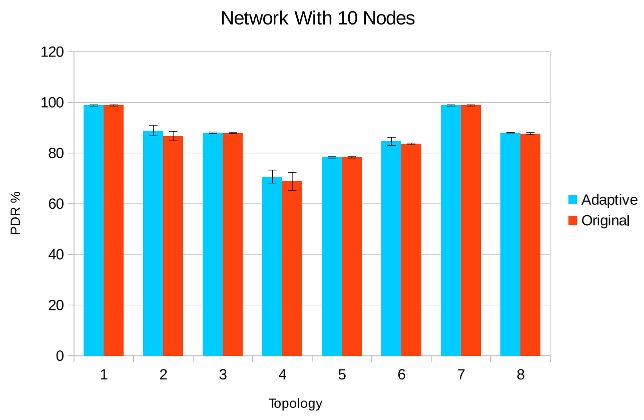


Fig. 5. Average PDR and confidence interval achieved with network of 10 nodes simulation.

B. *Methodology for Real-World Experiments*

To validate the performance of the adaptive protocol with real world network, a WSN was set up with 10 Zolertia Z1 nodes[3] running Contiki OS. In the network, the maximum hop distance was 2 and multiple runs were performed. As in the simulation, each node created and sent a packet in average every 10 seconds. To avoid collisions, each packet was transmitted at a randomized time within the 10 seconds duration. Experiments were carried out using both the adaptive protocol and the original CTP, for 8 hours each round.

To evaluate the performance of the overall network, the average number of Packet Delivery Ratio was considered.

VIII. Results

A. *Learning Parameters*

The values for the learning parameters were obtained by checking 80 learning parameter combinations and conducting 10 rounds of simulations for the same combination.The value range for each variable was selected after investigating several previous works in the literature such as [38], [39]. For ϵ, the values 0.1, 0.2, 0.4, 0.6, and 0.8, for γ the values 0, 0.3, 0.5, and 0.8 and for α the values 0.2, 0.4, 0.6, and 0.8 were tested to obtain the best combination with a randomly deployed 10-nodes network. Fig. 4 illustrates the PDR obtained for different learning parameters ϵ, γ, and α. The best packet delivery rate values of the network were obtained when γ is 0.3, ϵ is 0.1 and α is 0.2 as marked in Fig. 4.

B. *Performance of the Adaptive Parameter Algorithm*

The results obtained by two evaluation methods, simulations and the real-world experiments, are presented in the next two subsections.

[3]https://zolertia.io/

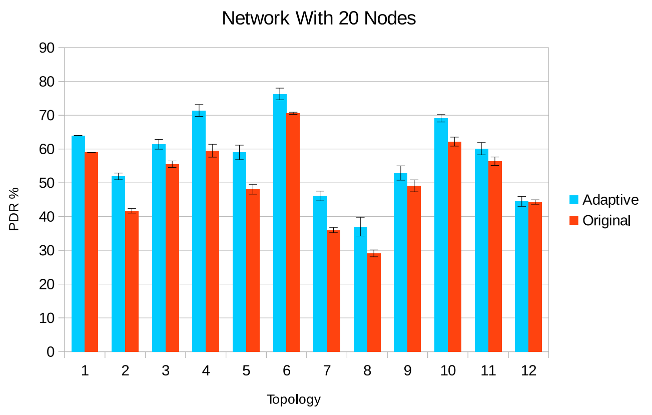


Fig. 6. Average PDR and 95% confidence interval achieved with network of 20 nodes simulation.

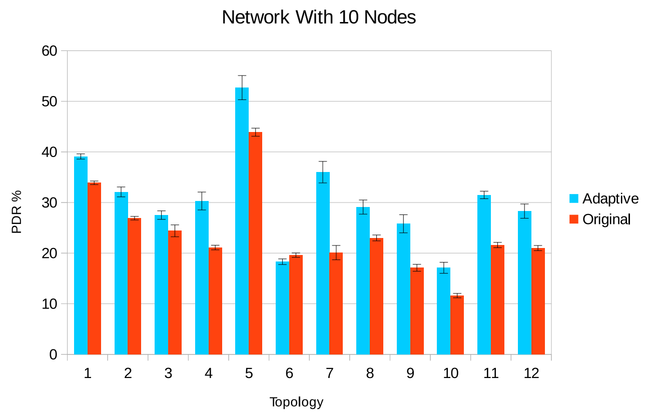


Fig. 7. Average PDR and confidence interval achieved with network of 30 nodes simulation.

1) Simulation Results: The average PDR values were calculated for the adaptive protocol and the original CTP. Figures 5, 6 and 7 illustrate the PDR and confidence interval of each network topologies with 10, 20, and 30 nodes respectively. The graphs show that the performance improvement gained by the adaptive protocol increases with the number of nodes in the topology. For the topologies with 10 nodes, the average performance improvement is 0.79%. But for the topologies with 20 and 30 nodes the average performance improvement is 13.44% and 29.41% respectively. Along with the number of nodes within the network, the amount of traffic that needs to be routed towards the sink increases. Also the number of packet collisions, network congestion, limitation of the queue of routing nodes impacts highly on the performance once the traffic is high. For this reason individual nodes identify their optimum parameter set according to their position in the network.

We furthermore analysed how many percent of the nodes changed their parameter settings away from the default settings, dependent on the number of the node's hop distance to the sink. The results given in Fig. 8 illustrate the number of nodes which have changed their parameter set away from the default for different hop counts. The results show that the

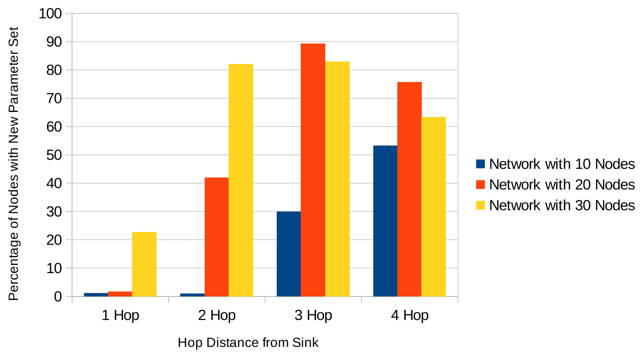


Fig. 8. Percentage of nodes from different hop distances, which changed their parameter set from the default value in simulated networks.

TABLE IV
COMPARISON OF PDR WITH DIFFERENT HOP DISTANCES.

Number of Nodes in the Network	Hop distance	Adaptive protocol	Original protocol
20	2	84.3%	63.5%
20	3	14.5%	3.9%
30	2	33.4%	16.2%
30	3	2.5%	0.23%

optimum parameter set differs from the default parameter set when the hop count is increased and the number of nodes increases in the network. Also their performance is tabulated in Table IV. With the PDR values it can be seen that Adaptive Protocol improves the performance with changed parameter sets.

Fig. 9 illustrates the PDR percentage at sink node for a network with 20 nodes. 10 different curves in the Fig. 9 illustrate 10 rounds of simulation results for the same topology. The convergence of the network is important to obtain a consistent performance. Hence the convergence time of each topology is analysed. All nodes within the 10-node topology converge to their optimum parameters after generating 500 packets respectively 5000 s. Most nodes in 10 nodes network, the original parameter set is identified as its best parameter set. Hence the required time for convergence is less. For networks with 20 nodes convergence occurs after 10000 seconds. The networks with 30 nodes converged between 10000 and 12000 seconds. Fig. 9 illustrates the convergence of the packet delivery rate after about 10000 packets when optimum parameter set is identified by each node. In most WSN applications long term deployment is required, so even spending 3 hours to identify the best parameter set will be beneficial to obtain a better performance of the network.

2) Results from Real World: Fig. 10 illustrates the packet delivery rate obtained with real nodes for the same topology in different rounds. The mean value obtained from the adaptive protocol and the original protocol are 94.6% and 81.4% respectively. Hence percentage improvement by the adaptive protocol is 16.21%. The confidence intervals for adaptive protocol and the original protocol are ±6.27% and ±11.73% respectively. The simulated scenarios had a static environment compared to the real world scenario. Due to the fact of changing environmental conditions, the real world scenarios

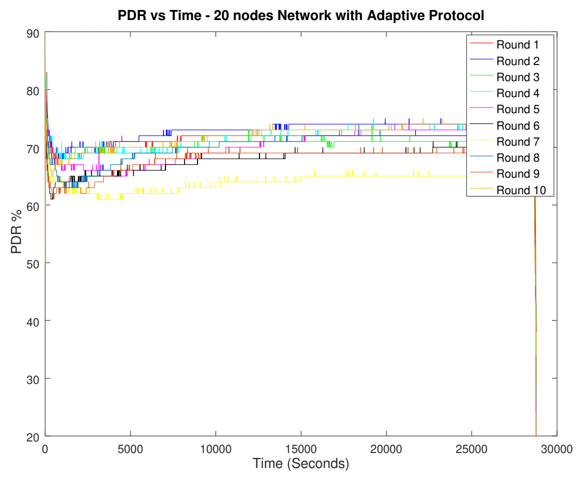


Fig. 9. Packet Delivery Rate Vs Time (s) in simulated 20 nodes network.

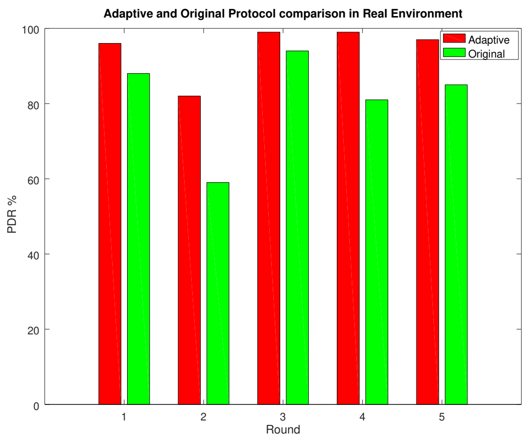


Fig. 10. PDR performance of real network with 10 nodes.

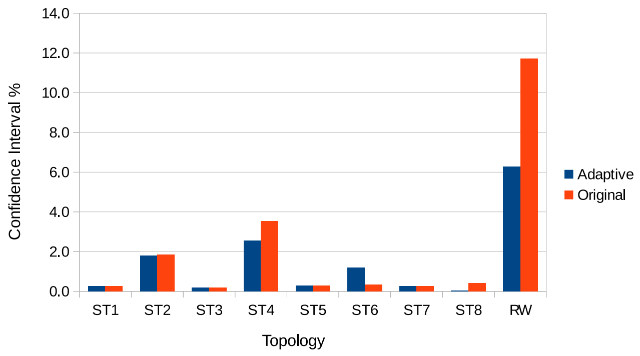


Fig. 11. Comparison of confidence interval in different simulation topologies and real world experiment.

have a higher variation in the PDR results. Fig. 11 illustrates the confidence interval in different simulation topologies and real world scenario. ST: Indicates the simulation topology and RW indicates the real world topology in Fig. 11. In real world setup none of the nodes were away from more than two hops from the sink. However 30% of nodes with a distance of two hops had parameter sets different from the original. Even in changing real world scenario, the adaptive algorithm improves the performance of the network by identifying optimum parameter values.

IX. Conclusion and Future Work

In this work we demonstrate that the individual performance of nodes in a network differs when a protocol with static parameter settings is deployed in different applications. For this reason we formulated an adaptive algorithm on top of the original protocol to learn from the environment and adapt the optimum parameter set to improve the performance of the network. The algorithm works in a distributed manner, hence an individual node evaluates its best parameter set without exchanging signalling packets with other stations. The optimum learning parameters were obtained using series of simulations with a network of 10 nodes. Using the best learning parameters we show that the formulated adaptive protocol performs above the original protocol using simulations and real world scenarios. The adaptive algorithm increases the performance over the original protocol. The identified solution might not be the optimum one, however Q learning is known to find a solution near to the optimum.

The performance of the adaptive algorithm is significant in networks with a higher number of nodes and networks with nodes placed several hops away from the sink node. We believe that the performance of large networks which are randomly placed will significantly improve with the introduced adaptive protocol.

In our work, the adaptive algorithm has been applied only on top of CTP. The best learning parameters, that is epsilon, alpha and gamma were evaluated using a network with 10 nodes. However this can be different for larger networks. Hence it is important to identify best learning parameters irrespective of the number of nodes in the network. As our future work we plan to implement the adaptive algorithm on top of another standard protocol with best learning parameters. For that initially local parameters which contributes to the performance significantly should be identified with their range of parameter values which can be vary. With the feedback of selected performance metrics, the parameters should be tuned.

References

[1] M. D. Andújar-Montoya, D. Marcos-Jorquer, F. M. García-Botella, and V. Gilrt-Ilesias, "A context-driven model for the flat roofs construction process through sensing systems, Internet-of-things and last planner system," *MDPI Sensors*, vol. 17, no. 7, p. 1691, 2017.

[2] M. Mafuta *et al.*, "Successful deployment of a wireless sensor network for precision agriculture in malawi," *SAGE Int. J. Distrib. Sensor Netw.*, vol. 9, no. 5, p. 150703, 2013.

[3] A. Ali, Y. Ming, S. Chakraborty, and S. Iram, "A comprehensive survey on real-time applications of WSN," *MDPI Future Internet*, vol. 9, no. 4, p. 77, 2017.

[4] M. Zimmerling, "Automatic parameter optimization of sensor network MAC protocols," Master's thesis, Institute of Systems Architecture, Technical University of Dresden, Germany, 2009.

[5] N. Z. binti Zubir, A. F. Ramli, and H. Basarudin, "Optimization of wireless sensor networks MAC protocols using machine learning; a survey," in *Proc. IEEE ICE2T*, 2017.

[6] A. Foerster and A. Foerster, *Emerging Communications for Wireless Sensor Networks*. London, UK: IntechOpen, 2011.
[7] M. A. Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 1996–2018, 2014.
[8] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection Tree Protocol," in *Proc. ACM SenSys*, 2009.
[9] V. Di Valerio, F. L. Presti, C. Petrioli, L. Picari, and D. Spaccini, "A self-adaptive protocol stack for underwater wireless sensor networks," in *Proc. IEEE OCEANS*, 2016.
[10] T. Dinh, Y. Kim, T. Gu, and A. V. Vasilakos, "An adaptive low-power listening protocol for wireless sensor networks in noisy environments," *IEEE Syst. J.*, vol. 12, no. 3, pp. 2162–2173, 2017.
[11] P. G. Park, C. Fischione, A. Bonivento, K. H. Johansson, and A. Sangiovanni-Vincentelli, "Breath: A self-adapting protocol for wireless sensor networks in control and automation," in *Proc. IEEE SECON*, 2008.
[12] T. Van Dam and K. Langendoen, "An adaptive energy-efficient MAC protocol for wireless sensor networks," in *Proc. ACM SenSys*, 2003.
[13] Y. Wang, W. Yang, R. Han, and K. You, "A network equivalent-based algorithm for adaptive parameter tuning in 802.15. 4 WSNs," *MDPI Sensors*, vol. 18, no. 7, p. 2031, 2018.
[14] P. Sun, G. Li, and F. Wang, "An adaptive back-off mechanism for wireless sensor networks," *MDPI Future Internet*, vol. 9, no. 2, p. 19, 2017.
[15] M. Steine *et al.*, "Proactive reconfiguration of wireless sensor networks," in *Proc. ACM MSWiM*, 2011.
[16] M. Abdelaal, P. Zhang, and O. Theel, "QoS improvement with lifetime planning in wireless sensor networks," in *Proc. IEEE MSN*, 2015.
[17] M. Steine, M. Geilen, and T. Basten, "A distributed feedback control mechanism for quality-of-service maintenance in wireless sensor networks," in *Proc. DSD*, 2012.
[18] M. Salayma, A. Al-Dubai, I. Romdhani, and M. B. Yassein, "Battery aware beacon enabled IEEE 802.15. 4: An adaptive and cross-layer approach," in *Proc. IEEE FedCSIS*, 2015.
[19] S. Brienza, M. Roveri, D. D. Guglielmo, and G. Anastasi, "Just-in-time adaptive algorithm for optimal parameter setting in 802.15. 4 WSNs," *ACM Trans. Auton. Adaptive Syst.*, vol. 10, no. 4, pp. 1–26, 2016.
[20] P. Park, P. Di Marco, C. Fischione, and K. H. Johansson, "Modeling and optimization of the IEEE 802.15. 4 protocol for reliable and timely communications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 3, pp. 550–564, 2012.
[21] G. M. Dias, M. Nurchis, and B. Bellalta, "Adapting sampling interval of sensor networks using on-line reinforcement learning," in *Proc. IEEE WF-IoT*, 2016.
[22] V. Di Valerio, F. L. Presti, C. Petrioli, L. Picari, D. Spaccini, and S. Basagni, "CARMA: Channel-aware reinforcement learning-based multi-path adaptive routing for underwater wireless sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 11, pp. 2634–2647, 2019.
[23] G. Kour and R. Saabne, "Real-time segmentation of on-line handwritten arabic script," in *Proc. IEEE ICFHR*, 2014.
[24] F. A. Aoudia, M. Gautier, and O. Berder, "RLMan: An energy manager based on reinforcement learning for energy harvesting wireless sensor networks," *IEEE Trans. Green Commun. Netw.*, vol. 2, no. 2, pp. 408–417, 2018.
[25] C. Savaglio, P. Pace, G. Aloi, A. Liotta, and G. Fortino, "Lightweight reinforcement learning for energy efficient communications in wireless sensor networks," *IEEE Access*, vol. 7, pp. 29 355–29 364, 2019.
[26] Y. Kosunalp, Selahattin ad Chu, P. D. Mitchell, D. Grace, and T. Clarke, "Use of Q-learning approaches for practical medium access control in wireless sensor networks," *Eng. Applicat. Artificial Intell., Elsevier*, vol. 55, pp. 146–154, 2016.
[27] A. Forster and A. L. Murphy, "FROMS: Feedback routing for optimizing multiple sinks in WSN with reinforcement learning," in *Proc. ISSNIP*, 2007.
[28] T. T. Le and S. Moh, "An energy-efficient topology control algorithm based on reinforcement learning for wireless sensor networks," *ORES Int. J. Control Autom.*, vol. 10, no. 5, pp. 233–244, 2017.
[29] W. Guo, C. Yan, and T. Lu, "Optimizing the lifetime of wireless sensor networks via reinforcement-learning-based routing," *SAGE Int. J. Distrib. Sensor Netw.*, vol. 15, no. 2, 2019.
[30] J. Yang, S. He, Y. Xu, L. Chen, and J. Ren, "A trusted routing scheme using blockchain and reinforcement learning for wireless sensor networks," *MDPI Sensors*, vol. 19, no. 4, p. 970, 2019.
[31] A. Dunkels, "Rime-a lightweight layered communication stack for sensor networks," in *Proc. EWSN*, 2007.
[32] S. Brienza, D. De Guglielmo, C. Alippi, G. Anastasi, and M. Roveri, "A learning-based algorithm for optimal MAC parameters setting in IEEE 802.15. 4 wireless sensor networks," in *Proc. ACM PE-WASUN*, 2013.
[33] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, Massachusetts, USA: MIT Press, 2018.
[34] A. D. Tijsma, M. M. Drugan, and M. A. Wiering, "Comparing exploration strategies for Q-learning in random stochastic mazes," in *Proc. IEEE SSCI*, 2016.
[35] J. Vermorel and M. Mohri, "Multi-armed bandit algorithms and empirical evaluation," in *Proc. ECML*, 2005.
[36] A. Velinov and A. Mileva, "Running and testing applications for Contiki OS using Cooja simulator," in *Proc. ITRO*, 2016.
[37] F. Österlind, "A sensor network simulator for the Contiki OS," *SICS Research Report*, 2006.
[38] S. Kosunalp, "A new energy prediction algorithm for energy-harvesting wireless sensor networks with q-learning," *IEEE Access*, vol. 4, pp. 5755–5763, 2016.
[39] M. Chincoli and A. Liotta, "Self-learning power control in wireless sensor networks," *Sensors*, vol. 18, no. 2, p. 375, 2018.

Piumika Karunanayake is a Senior Lecturer at the Department of Electrical, Electronic and Telecommunication Engineering, Sir General Kotelawala Defence University, Sri Lanka. She has obtained BSc. Hons and MSc.in Electronics and Telecommunication Engineering from University of Moratuwa, Sri Lanka. Currently she is reading for her PhD at University of Bremen, Germany. Her research interests are in wireless sensor networks, machine learning and vehicular networks.

Andreas Könsgen works as a Postdoctoral Researcher at the Sustainable Communication Networks Working Group at University of Bremen since 2009. He has obtained his Diploma degree in Electrical Engineering at Aachen University of Technology, Germany, and his PhD degree at University of Bremen. His current research interest is the deployment of machine learning in different network architectures such as in sensor networks or in the area of future Internet.

Thushara Weerawardane is working as Senior Lecturer at the Department of Computer Engineering, Sir General Kotelawala Defence University, Sri Lanka. He has obtained BSc. Hons in Electrical Engineering, University of Moratuwa, Sri Lanka, MSc in Information & Communication Technology and PhD in Mobile Communication, University of Bremen, Germany. His research interests are in wireless sensor networks, Internet of things, 5G networks, machine learning, data science and statistical analysis.

Anna Förster is currently a Professor at the University of Bremen in Germany. She has obtained her MSc degree in Computer Science from the Free University of Berlin in 2005 and her PhD from the University of Lugano in 2009. Her research interests lie in the areas of opportunistic networks, wireless sensor networks for challenged environments and network simulation. She is especially interested in applications of ICT to achieve the sustainable developments goals.