Q-Learning Enhanced Gradient Based Routing for Balancing Energy Consumption in WSNs

Bazyli Debowski, Petros Spachos, Shawki Areibi
School of Engineering
University of Guelph, Guelph, ON, Canada
Email: {debowskb, petros, sareibi}@uoguelph.ca

***Abstract*—Energy is a sparse and valuable resource in Wireless Sensor Networks (WSNs). Using it efficiently and effectively can mean longer node and network lifetimes with less reliance and strain on energy harvesting components. In this work we focus on optimizing energy consumption in WSNs at the network layer. A gradient based routing protocol is proposed, that integrates a Reinforcement Learning (RL) component to learn and seek out routes which deplete node energy in a balanced manner. This RL enhanced protocol was compared against three other gradient based protocols, including a greedy, shortest-paths variation serving as a baseline. All protocols were simulated under conditions of heavily imbalanced network load. Based on the simulation results, the network lifetime of the RL enhanced protocol was nearly doubled in comparison to the baseline protocol, while average packet delay was significantly reduced.**

I. Introduction

Wireless Sensor Networks (WSNs) are powered by batteries which deplete in energy reserves and must be replaced periodically. By consuming energy in an efficient and effective manner the lifetime for individual nodes as well as for the network as a whole can be extended. The alternative is to maintain nodes and replace batteries more frequently, or to use larger and higher cost batteries.

Even in WSNs where energy harvesting technology is used, it is of benefit to ensure energy is consumed efficiently and effectively in order to reduce the reliance and strain on energy harvesting components. Determining how energy is consumed in a WSN can be tackled at various levels of consideration (e.g. hardware, physical layer, mac layer, network layer, etc.). In this work we have approached the problem at the level of the network layer of the protocol stack.

A routing protocol was developed that is a hybrid of gradient based routing [1] and a form of Reinforcement Learning (RL) known as Q-Learning [2]. The proposed protocol is coined Q-Smart Gradient based routing (QSGrd). By using a transmission gradient, the following major benefits are realized: i) nodes do not need to know their geographic coordinates, ii) shortest paths to sink can be established quickly and easily maintained [1], and iii) the Q-Learning component is supplied a strong starting point and spends less time converging. By incorporating a Q-Learning component, the following further major benefits are also realized: i) higher level, abstract routing goals can be defined and realized by changing only the reward function [3], and ii) routes change dynamically and continuously in response to changing network conditions.

Three other gradient based routing protocols that *are not* RL enhanced, including a greedy, shortest-paths variation serving as a baseline, were also developed and tested. These other protocols track our logical progression towards the design of the QSGrd protocol proposed. They also serve to support the validity of the transmission gradient approach implemented.

The remainder of this paper is structured as follows. In Section II a brief overview is presented on Q-Learning and how it has been used for WSN routing in the past. In Section III a detailed description is given of the routing protocols, the simulation framework, and the experiments with which the protocols were tested. In Section IV results and discussion are presented. In Section V conclusions are given and future work is suggested.

II. Related Work

The QSGrd protocol proposed in this work is a unique hybrid combination of gradient based routing and Q-Learning/Q-Routing techniques. For an introduction to gradient based routing the reader is referred to [1]. For an introduction to Q-Learning and how it can be applied to solve routing problems, the reader is referred to [2] and [4].

Q-Learning has been used extensively in WSN routing to achieve various routing goals. The simplicity of the algorithm and its model-free nature fit well with the constrained resources of WSN nodes. Also, the nature of WSN communication is such that transmissions are broadcast and can be overheard by all 1-hop neighbours. This fits well with the need to frequently update neighbours with up-to-date Q-Values.

The work in [4] was one of the first to introduce a Q-Learning based routing protocol (commonly referenced as Q-Routing). The authors used Q-Learning to determine shortest paths in a wired network based on the packet delay. As congestion grew in specific areas of the network, Q-Routing was able to find routes that avoided the congested areas, while geographic shortest path routing was not able to adapt.

In [3], the authors presented Adaptive Tree Protocol (ATP). They used a spanning tree based protocol as a foundation and enhanced it with multi-objective Q-Learning. They indicated that using Q-Learning enabled thinking about routing at a higher logical level of abstraction. As evidence of this concept they implemented and tested two versions of their protocol, differing only in the reward/objective function used for the learning component.

In [5], the authors used Q-Learning to augment a protocol based on nodes knowing their geographic locations. The Q-Learning component was used to learn values for expected number of retransmissions. We use a similar measure which we refer to as the estimated average number of transmissions. It is of value to observe from [5] that Q-Learning can be used to augment a much larger protocol/model by learning only a small component of information used in the larger whole.

In [6], the authors augmented a geographic routing protocol with Q-Learning in order to: i) add the ability to avoid network holes and dead nodes, and ii) balance energy consumption in the network by considering node energy. This work is similar to ours in that the Q-Learning component considered energy and progress to sink, and the network lifetime was approximately doubled. However, a key difference is that in [6] the protocol required nodes to know their geographic location.

In [7], the authors introduced QELAR; a protocol for routing in under water sensor networks. One unique idea in this Q-Learning based approach was to consider not only node energy, but also average node energy for node neighbourhoods. This work covered in great detail the impact of various parameter adjustments in the authors' protocol. We have drawn inspiration from the experiments the authors performed and the way they documented and presented their results.

The QSGrd protocol proposed in this paper differs from all of these Q-Learning based protocols primarily in its hybridization of a transmission gradient with a Q-Learning component. In QSGrd, the transmission gradient encodes the information on shortest paths to sink, while the Q-Learning component improves these routes so as to both balance energy consumption and to reduce packet delay. By combining these two techniques, their strengths are enhanced while their weaknesses are diminished.

III. Methodology

A. Simulation Framework

In this section the simulation framework developed in OMNeT++ is presented. This framework serves as the test bed where the routing protocols and simulations took place. The logical structure of this framework is best represented as a 4-layer protocol stack as shown in Fig. 1. We describe this framework in a bottom-up manner, beginning with the physical layer.

The **physical layer** is responsible for transmitting and receiving all network packets. There are 3 types of packets: i) Data Frames, ii) ACK Frames, and iii) Status Frames. Data Frames are meant to represent payloads in network traffic, whereas ACK Frames and Status Frames are a type of meta-data used to control network traffic. The physical layer is also responsible for modelling transmission errors. It follows a probability function for Packet Error Rate (PER) as specified in (1). This implementation of PER is similar to the one found in [5].

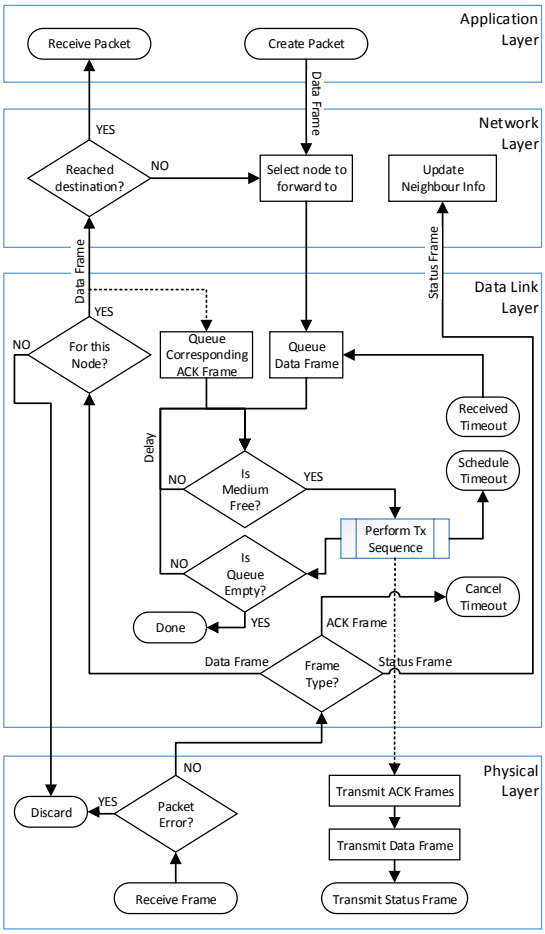

Fig. 1. Logical structure of 4-Layer protocol stack implemented.

Let R be the maximum possible transmission range of a node (set to 100 m in our experiments). Then the probability, P_{err}, of a transmission error occurring is given by

$$P_{err} = \begin{cases} 0.5 \times \left(\frac{d}{0.75R}\right)^{\kappa}, & \text{if } d \leq 0.75R \\ 1 - 0.5 \times \left(2 - \frac{d}{0.75R}\right)^{\kappa}, & \text{if } 0.75R < d \leq R \end{cases} \quad (1)$$

where d is the Euclidean distance between the two communicating nodes and κ is the attenuation factor (set to a value of 7 in our experiments). Nodes with more than R distance between them are considered unable to communicate, hence P_{err} is not modelled beyond this range (i.e. P_{err} is 100% when $d > R$). This PER model applies *only* to Data Frames. Status Frames and ACK Frames are assumed guaranteed to arrive if $d \leq R$.

The **data link layer** is primarily responsible for: i) queuing outgoing Data Frames and ACK Frames, ii) preventing packet collisions, iii) executing a transmission sequence when the medium is free, and iv) re-transmitting any Data Frames for which it has not received an ACK Frame. Each node holds a queue for both outgoing Data Frames and outgoing ACK Frames. Each time a node receives a Data Frame addressed to itself, the node queues a corresponding ACK Frame addressed to the node from which it received the Data Frame. Sending

such an ACK Frame to its neighbour informs the neighbour that this node has successfully received the Data Frame that was sent to it.

If either the ACK Frame queue or the Data Frame queue is *not* empty, the node polls the transmission medium for a chance to transmit. It continues to poll the medium, with a short delay, until it detects the medium to be free. The transmission medium is considered to be free if no nodes within $2 \times R$ are currently transmitting.[1] At this point the node begins a transmission sequence. In this way, collisions are prevented in the framework. A single transmission sequence performed by a node entails transmitting: i) *All* queued ACK Frames, ii) *One* queued Data Frame, and iii) *One* Status Frame (generated at the time of transmission).

Whenever a node completes a transmission sequence it begins a timeout timer corresponding to the Data Frame that was sent out. If the node does not receive an ACK Frame from its neighbour within a predefined time limit, acknowledging that the neighbour has received the Data Frame, a timeout event occurs. At this point the node assumes there must have been a transmission error when sending the Data Frame. It then re-queues this Data Frame to be re-transmitted later.

Routing decisions take place at the **network layer**. When a node receives a Data Frame addressed to itself and it is not the final destination of the frame received, it uses the routing protocol specified to determine which neighbour to forward the packet to. The network layer uses information provided to it in the Status Frames it receives to maintain up-to-date information about neighbouring nodes. This information is in turn, used by the network layer to make forwarding decisions according to the rules imposed by the routing protocol.

Data Frames are generated and consumed at the **application layer**. When a Data Frame reaches its final destination node, it is passed up to the application layer where it is then recorded. Each node is capable of generating Data Frames in the application layer, and will do so at the rate specified by the simulation parameters. Data Frames generated here are then passed down to the network layer for forwarding.

B. Routing Protocols

In total 4 routing protocols were implemented and tested in this work:

1) Greedy Gradient (GGrd)
2) Energy Aware Gradient (EAGrd)
3) Energy Smart Gradient (ESGrd)
4) Q-Smart Gradient (QSGrd)

For each protocol implemented the ultimate goal is to determine the best neighbouring node to forward a Data Frame to. A transmission gradient was used as the basis for all of the protocols. The GGrd protocol serves as a baseline for comparison with all other protocols. The EAGrd and ESGrd protocols build upon the GGrd protocol. Their evolution tracks the logical progression of the design towards the final formulation of the QSGrd protocol.

An example of the transmission gradient established and used by each routing protocol is shown in Fig. 2. The value at each node represents the estimated average least number of transmissions required to reach the base station from that node. The connecting edges between nodes represent shortest paths. The value at each edge represents the probability of successfully transmitting a Data Frame between the two nodes connected by that edge. This probability is determined by

$$P_{suc}(d(n_{src}, n_i)) = 1 - P_{err}(d(n_{src}, n_i)) \tag{2}$$

where n_{src} is the node forwarding the packet, n_i is a neighbour of n_{src}, $d(n_{src}, n_i)$ is the Euclidean distance between n_{src} and n_i, and P_{err} was defined in (1).

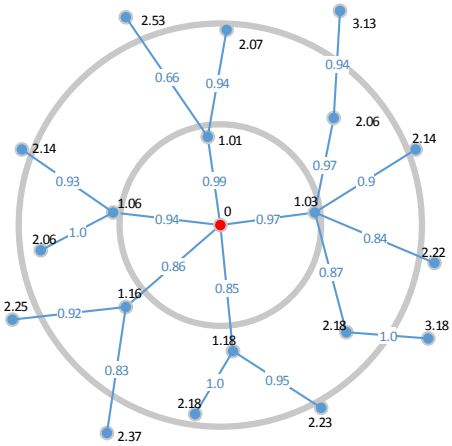

Fig. 2. Example of transmission gradient used.

The transmission gradient is constructed in a manner resembling dynamic programming. Specifically, it is constructed according to the algorithm outlined in Fig. 3. In this algorithm, the "estimateProbabilityTxSuccess()" function estimates transmission success between two nodes, as defined in (2).

The baseline protocol implemented is the Greedy Gradient protocol (GGrd). The GGrd protocol makes forwarding decisions by each time selecting the neighbouring node which results in the most progress towards the base station. The progress measured includes considering the probability of successfully transmitting to the neighbour selected. Details for this process are presented in Fig. 4.

The Energy Aware Gradient protocol (EAGrd) builds on the GGrd protocol by considering also the residual energy of nodes. When making forwarding decisions, the progress to base station for any neighbouring node selected is taken to include the residual energy of that node. In this way, as energy is drained along shortest paths, the protocol begins to select nodes with higher energy levels along longer paths.

The Energy Smart Gradient protocol (ESGrd) builds on the EAGrd protocol by also considering nodes that result in little or no progress to base station, *iff* those nodes have considerably more energy than the node forwarding the Data

[1]Although this form of collision avoidance is in a way overly optimistic (nodes could not in reality sense into $2 \times R$), it is also in ways overly conservative. It is conservative because it prevents *all* collisions, even those that would occur at nodes to which no packets were addressed.

```
ConstructGradient:
Variables:
sink // The sink node (base station)
node // Node where ReceivedStatusFrame event triggered
sf   // A Status Frame containing sending node's
       estimated number of transmissions to sink
n.numTxToSink // Avg. min number of transmissions
                 required to reach sink from n
-----------------------------------------------------
// Initialize the process at the sink node
sink.numTxToSink ⟵ 0
sink.broadcastStatusFrame()

// Any node receiving a Status Frame does following
OnEvent: ReceivedStatusFrame do
  select n in node.neighbours where n = sf.sender do
    n.numTxToSink ⟵ sf.senderNumTxToSink
    pSuccess ⟵ estimateProbabilityTxSuccess(node, n)

    // e.g. if pSuccess = 0.33 then node requires on
    // average 1 / 0.33 = 3 transmissions to reach n
    numTxToN ⟵ 1 / pSuccess

    // If node's best known numTxToSink has changed it
    // broadcasts a Status Frame to its neighbours
    if n.numTxToSink + numTxToN < node.numTxToSink do
      node.numTxToSink ⟵ n.numTxToSink + numTxToN
      node.broadcastStatusFrame()
    end
  end
end
```

Fig. 3. Algorithm: ConstructTransmissionGradient

```
SelectNodeToForwardTo_GGrd():
Variables:
source         // The node forwarding the packet
neighboursList // List of forwarding node's neighbours
n.numTxToSink  // Estimated number of transmissions
                  required to reach sink from node n
-----------------------------------------------------
bestProgress ⟵ 0
nodeToForwardTo ⟵ source

for each n in neighboursList do
  txProgress ⟵ source.numTxToSink - n.numTxToSink
  pSuccess ⟵ estimateProbabilityTxSuccess(source, n)
  progress ⟵ pSuccess x txProgress
  if progress > bestProgress do
    bestProgress ⟵ progress
    nodeToForwardTo ⟵ n
  end
end

return nodeToForwardTo
```

Fig. 4. Algorithm: SelectNodeToForwardTo_GGrd

Frame. In this way, Data Frames are routed into less used and thus higher energy regions of the network.

The Q-Smart Gradient protocol (QSGrd) builds upon the GGrd protocol in a similar fashion to EAGrd and ESGrd. However, it does so by incorporating a Q-Learning component. This Q-Learning component considers both the estimated average least number of transmissions to base station at each node (as determined by the established transmission gradient), as well as the residual energy level at each node. In this way, the protocol is able to learn routes that minimize route length and maximize residual energy of the nodes along those routes.

Details on how the QSGrd protocol operates are as follows. Each node maintains a Q-Value for itself (which is included in the Status Frame packets that are broadcast to neighbours). This Q-Value is a measure of how useful this node is as judged by the metrics of the Q-Learning reward function. When deciding which neighbouring node to forward a packet to, the forwarding node will select the neighbour that results in the highest new Q-Value for itself. The QSGrd algorithm for selecting the neighbouring node to forward to is presented in Fig. 5. In the algorithm the function qValueIfSelect(n) performs the calculations specified by (4) to (7). The assignment operation at the end of the algorithm performs the calculations specified by (3).

```
SelectNodeToForwardTo_QSGrd():
Variables:
source         // The node forwarding the packet
neighboursList // List of forwarding node's neighbours
n.qValue       // Node's last calculated Q-Value
a              // Learning rate alpha
-----------------------------------------------------
bestQValue ⟵ -1
nodeToForwardTo ⟵ source

// Select neighbour yielding highest new qValue
for each n in neighboursList do
  qNew ⟵ qValueIfSelect(n)
  if qNew > bestQValue do
    bestQValue ⟵ qNew
    nodeToForwardTo ⟵ n
  end
end

// Update Q-Value of source based on decision outcome
// Later transmitted to neighbours via Status Frame
qOld ⟵ source.qValue
source.qValue ⟵ (1 - a) x qOld + a x bestQValue

return nodeToForwardTo
```

Fig. 5. Algorithm: SelectNodeToForwardTo_QSGrd

All Q-Values are initially set to 1. Once learning begins Q-Values are determined according to the mathematical operations specified by (3) to (7). Equation (7) determines the reward value for progress made towards sink. Similarly, (6) determines the reward value for energy improvement. In (7), actions (a_n) are rewarded for reducing the number of remaining transmissions for the packet to reach the base station. In (6), actions (a_n) are rewarded for increasing the residual energy of the node holding the packet. For both (6) and (7) a sigmoid function is applied so that: i) rewards are always positive, and ii) there is a behaviour of diminishing gains and diminishing losses as the delta values move away from 0. Equation (5) performs a weighted combination of (6) and (7) allowing adjustment of which variable (energy or progress) is favoured more heavily. Equation (4) is the discounted reward function, central to Q-Learning (see [2]). We incorporate into (4) the $P_{suc}(a_n)$ term so that reward values are scaled by how achievable the corresponding action really is. Equation (3) applies the learning rate.

$$Q_s = (1 - \alpha) \times Q_s + \alpha \times Q_{new} \tag{3}$$

$$Q_{new} = (\gamma \times Q_n + (1 - \gamma) \times R(a_n)) \times P_{suc}(a_n) \tag{4}$$

$$R(a_n) = \omega_e \times R_e(a_n) + (1 - \omega_e) \times R_p(a_n) \tag{5}$$

$$R_e(a_n) = sigm(\Delta E(a_n) \times \pi \times \beta_e) \tag{6}$$

$$R_p(a_n) = sigm(\Delta T_{est}(a_n) \times \pi \times \beta_p) \tag{7}$$

where α is the learning rate, γ is discounted reward factor, Q_s is the Q-Value of the forwarding node s, Q_n is the Q-Value of the neighbour n being considered, a_n is the action of forwarding the packet to node n, $P_{suc}(a_n)$ is the probability of successfully performing action a_n, ω_e is the weight of energy considerations, $\Delta E(a_n)$ is the energy level difference between node s and node n where energy level is a value between 0 and 1, $\Delta T_{est}(a_n)$ is the difference between node s and node n in the estimated number of transmissions to the sink as observed in the network gradient, β_e is an energy difference scaling factor, and β_p is a transmission difference scaling factor.

IV. Results

A. Experimental Setup

The aim of the experiments performed was to simulate a plausible and meaningful network scenario that was general enough to be widely applicable. Specifically, what was desired was to test conditions of unequally distributed network load. The scenario of an outdoor environmental monitoring WSN was chosen as a fitting application and was loosely simulated. The following assumptions were made:

- There was a single sink located at the centre of the network, referred to as the base station. This base station was taken to have infinite battery capacity.
- Nodes were placed in a square field in a grid-like pattern where each node was assigned to a single cell in the grid. Each cell contained exactly one node. The precise location of a node within such a cell was randomly chosen with a circular and centre-weighted probability distribution.
- The exact layout configuration tested was a $1400\ m \times 1400\ m$ field divided into a 31×31 cell grid. This resulted in a total area of just under $2\ km^2$ and total node count of just under 1000 nodes. With this layout the average distance between nodes on the horizontal and the vertical was $47\ m$. On the diagonal average distance was $66\ m$.
- Nodes were assumed to have a transmission range of up to $100\ m$. Using the PER model defined by (1), the probability of transmission success was 0.5 at a range of $75\ m$. Using an attenuation factor $\kappa = 7$, probability of success at $d = 66\ m$ (average diagonal node separation) was $P_{suc} \approx 0.80$ and at $47\ m$ (average horizontal node separation) was $P_{suc} \approx 0.98$.
- Tests were configured so as to terminate the simulation on the first node failure (when the energy level of any node dropped below 0.001 of full capacity).
- The energy costs for transmitting and receiving data were chosen such that the cost for transmitting was slightly higher than the cost for receiving. The exact values chosen were $15\ mW$ and $10\ mW$ respectively. The battery capacity for each node was intentionally specified to be very small at a value of $1\ mW \cdot h$. Nodes were assumed to consume energy only during transmitting and receiving data.
- All nodes were assumed to transmit data periodically (except the base station) at a relatively low frequency to the sink. The exact value tested was 1 Data Frame per 100 seconds per node. To simulate unbalanced network load, a small North West region of the network was configured to transmit data at a relatively much higher rate. Specifically, nodes$_{(3,3)(3,4)(3,5)(4,3)(4,4)(4,5)(5,3)(5,4)and(5,5)}$ each transmitted 2 Data Frames per second, where node$_(i,j)$ is located at row i and column j of the grid.
- Nodes were assumed to have a Data Frame buffer able to hold up to 16 Data Frames. The Data Frame generation rates were chosen, after some initial experimentation, to be low enough such as to avoid the scenario where packets might be dropped at receiving nodes due to buffers being full.

TABLE I
Simulation Parameters

Parameter	Value
Transmit Power	$15\ mW$
Receive Power	$10\ mW$
Battery Capacity	$1\ mW \cdot h$
Transmit Rate	$250\ kbps$
Data Frame Size	$1000\ bits$
ACK Frame Size	$50\ bits$
Status Frame Size	$50\ bits$
Buffer Length	16 Data Frames
Data Frame Generation Rate	2 and 0.01 fps
Transmission Range	$100\ m$
Attenuation Factor κ	7
Max Retransmissions Per Hop	5
α	0.2
γ	0.7
ω_e	0.65
β_e	2.0
β_p	0.5

A complete list of all simulation parameter values used are given in Table I. The simulation was tested under the parameters and conditions specified, once for each routing protocol using the same random seed. Thus the network topology was identical for each test and for each routing protocol. The simulation was set to run until the first occurrence of a node failure was detected. Several metrics were measured during simulation run time at $50\ s$ intervals. We present our findings in the following section.

B. Results and Analysis

In Fig. 6 minimum node energy vs. simulation time is plotted for each routing protocol. The minimum node energy is observed as the lowest residual energy level of any node in the entire network. This metric provides a measure of how well energy consumption is being distributed in the network. In Fig. 6 it can be seen that the energy balancing protocols track a shallower slope than the baseline protocol and thus lead to longer network lifetime. The ESGrd and QSGrd protocols greatly outperform the EAGrd protocol in increasing minimum

node energy and extending network lifetime. The curving nature of the plot for QSGrd indicates that the protocol put more emphasis on balancing energy consumption as the simulation elapsed. Given this result, we believe that in future work it might be worthwhile to experiment with dynamic parameter values for the QSGrd protocol (e.g. changing weight of energy consideration during simulation run time).

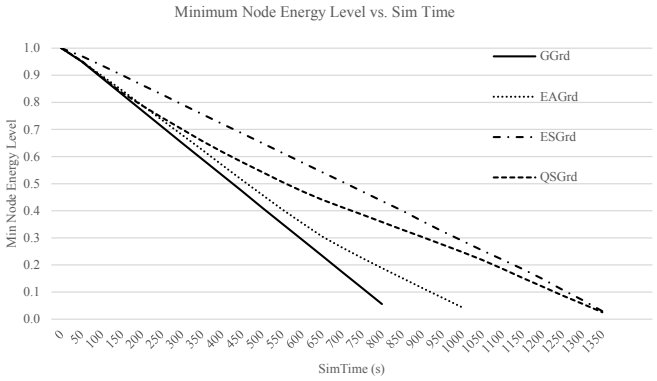

Fig. 6. Minimum node energy vs. sim time for each protocol.

In Fig. 7 average packet delay vs. simulation time is plotted for all packets travelling through the network. QSGrd yielded significantly lower packet delay times despite the longer routes travelled by this protocol. This indicates that the main source of packet delay in these results is not the number of transmissions, but rather the number of re-transmission due to packet delivery error. When there is a packet delivery error the sending node must wait for a timeout to occur before re-transmitting the packet. The duration of the timeout timer is relatively long compared to transmission times and time spent contending for the medium. Thus these results indicate that QSGrd is able to learn routes with higher link quality and lower PER, and thus lower packet delay. This behaviour is attributed to the $P_{suc}(a_n)$ term incorporated into (4).

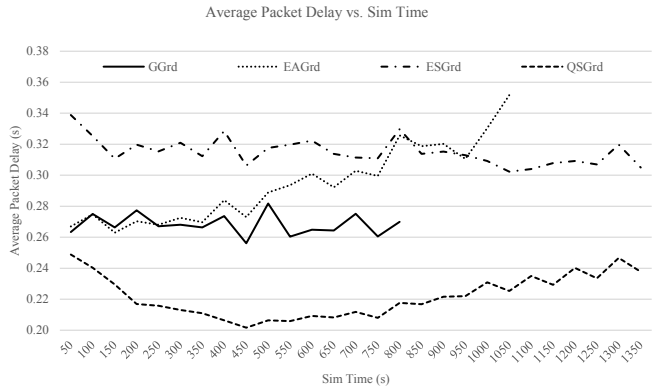

Fig. 7. Average packet delay vs. sim time for each protocol.

Table II summarizes the main observations for this work. It can be seen that ESGrd and QSGrd both provided over $1.5\times$ increase in number of packets delivered and in network lifetime.[2] QSGrad achieved this while also maintaining a significantly lower packet delay than all other protocols tested (including GGrd which focussed purely on shortest paths).

TABLE II
NETWORK LIFETIME

	GGrd	EAGrd	ESGrd	QSGrd
Total Runtime (s)	837.8	1045.4	1375.3	1382.1
Lifetime Increase	1x	1.25x	1.64x	1.65x
Packets Received	22960	28353	37459	38021
Packets Increase	1x	1.23x	1.63x	1.66x

V. CONCLUSION

In this work a Q-Learning enhanced, gradient based routing protocol coined QSGrd was introduced. The network performance of QSGrd was compared with three other gradient based routing protocols. Based on simulation results, QSGrd was determined to be the most powerful and effective of all protocols tested. A close performer was ESGrd, which has similar implementation to QSGrd albeit without a Q-Learning enhancement. Both protocols greatly extended network lifetime by effectively balancing energy consumption, but QSGrd was able to do so while also decreasing packet delay.

The results show the power and flexibility of the Q-Learning technique and the possibility to enhance existing protocols with Q-Learning. For future work, it would be useful to experiment with dynamic parameter values in the Q-Learning component. It would also be useful to determine what other routing goals can be effectively achieved when hybridizing gradient based routing with Q-Learning.

REFERENCES

[1] T. Watteyne, K. Pister, D. Barthel, M. Dohler, and I. Auge-Blum, "Implementation of gradient routing in wireless sensor networks." in *GLOBECOM*, 2009, pp. 1–6.

[2] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[3] Y. Zhang and Q. Huang, "A learning-based adaptive routing tree for wireless sensor networks," *Journal of Communications*, vol. 1, no. 2, pp. 12–21, 2006.

[4] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," *Advances in neural information processing systems*, pp. 671–671, 1994.

[5] R. Arroyo-Valles, R. Alaiz-Rodríguez, A. Guerrero-Curieses, and J. Cid-Sueiro, "Q-probabilistic routing in wireless sensor networks," in *Intelligent Sensors, Sensor Networks and Information, 2007. ISSNIP 2007. 3rd International Conference on*. IEEE, 2007, pp. 1–6.

[6] S. Dong, P. Agrawal, and K. Sivalingam, "Reinforcement learning based geographic routing protocol for uwb wireless sensor network," in *Global Telecommunications Conference, 2007. GLOBECOM'07. IEEE*. IEEE, 2007, pp. 652–656.

[7] T. Hu and Y. Fei, "Qelar: a machine-learning-based adaptive routing protocol for energy-efficient and lifetime-extended underwater sensor networks," *Mobile Computing, IEEE Transactions on*, vol. 9, no. 6, pp. 796–809, 2010.

[2]EAGrd, ESGrd, and QSGrd each perform more computations than the baseline GGrd protocol. The energy expended during these computations was not considered in this work. A more accurate estimate of energy savings can be obtained by modelling the energy consumed by nodes during processing. For QSGrd, (4) to (7) are computed once for each neighbouring node in range.