Balancing Energy Expenditure in WSNs through Reinforcement Learning: A Study

Anna Förster
University of Lugano, Switzerland
Email: anna.egorova.foerster@lu.unisi.ch

Amy L. Murphy
FBK-IRST, Trento, Italy
Email: murphy@fbk.eu

***Abstract*—This work describes a study of applying reinforcement learning to balance energy expenditure in wireless sensor networks, contributing both with a protocol and a novel observation regarding exploration in reinforcement learning. Our starting point is FROMS, our own multi-source multi-sink routing protocol which exploits learning to identify the lowest cost paths for data delivery. The primary modification here is an extension of the cost function to consider node battery levels in addition to our previous use of path length. The result is a dynamic cost function that greatly affects the behavior of the underlying reinforcement learning approach. This paper provides two main contributions. First, it describes an extension to FROMS that effectively balances the energy expenditure across all network nodes. Second, and most importantly, it offers a study of the effects of a dynamic cost function in a reinforcement learning setting. Specifically, we show that explicit route exploration becomes unnecessary, and instead the dynamic nature of the cost function forces a new form of implicit exploration which sufficiently and efficiently learns the best data paths for balancing energy.**

I. INTRODUCTION

One of the dominant features of wireless sensor network nodes is their limited energy reserves, driving research into energy efficient protocols, frameworks and approaches. Work in this field concentrates either on a single network layer (e.g., MAC, network routing, or application) or exploits cross-layer design. Multiple cost metrics measure energy expenditure, either in terms of network lifetime or the balance of energy usage across nodes over the network lifetime. Traditional approaches measure the latency of packet arrival, noting that low latency implies few hops, which further implies low energy. Alternately, some methods consider routes with the minimal total energy expenditure per packet, considering nodes with variable transmission distances and hence energy consumption. However, such approaches tend to find a single best route and then deplete its energy before searching for alternates. This is addressed by newer approaches which include remaining node battery in their cost metrics [1].

Recently, many researchers have turned their attention to machine learning and its applicability in wireless ad hoc networks [2]. This family of algorithms enables autonomous and flexible network behavior and is briefly surveyed in Section II. To evaluate route effectiveness, these works typically use traditional, simple cost measures such as aggregation rate [3] or lowest latency [4]. Such cost metrics, however, are static; they do not change significantly throughout the network lifetime. Instead, sink mobility, node and communication failures, all dynamic cost metrics, greatly affect the behavior of the traditional reinforcement learning mechanisms with unknown practical significance. To the best of our knowledge, this paper represents the first study of the properties and behavior of reinforcement learning with changing costs in wireless networks.

We approach this through a study of our own reinforcement learning approach for multi-source, multi-sink routing, FROMS [5], to which we introduce a dynamic cost metric. Essential FROMS background is provided in Section III. Section IV shows how to design and implement cost functions based on a combination of the number of hops and remaining battery power to achieve a variety of optimization goals such as giving preference to overall network energy balance over per-packet costs. Finally we evaluate four functions through a simulation study in Section V. Our most important observation, and a key contribution of this paper, is that by using a highly dynamic cost function, such as ours based on constantly changing battery levels, the explicit exploration typically found in reinforcement learning algorithms becomes unnecessary. Instead, the dynamic nature of the function forces a kind of implicit exploration, leading to a protocol which is more energy-stable, predictable, and easier to understand than a typical reinforcement learning based approach. This key observation must be taken into consideration in the future design and evaluation of new algorithms based on machine learning for any environment with dynamic characteristics, especially WSNs.

II. RELATED WORK

Here we consider two specific topics related to the main contributions of this paper: energy-aware and reinforcement learning-based routing protocols for WSNs. Also, relevant metrics for energy expenditure and spreading and the design of cost functions are discussed.

Traditional routing protocols essentially try to minimize energy expenditure per packet. On one side this leads to shorter, very energy-efficient routes. However, these routes are quickly depleted and the network could become disconnected. Therefore, other research efforts additionally take into account the node's residual energy. Such approaches work in one of two ways: considering strictly localized information where only the neighbors' remaining energy is given [6]–[9] or full

global information where all remaining power levels for all nodes are known at the base station [10].

GRE-DD [6] and *LMMER* [7] belong to the first class and are both extensions of Directed Diffusion [11]. They consider the remaining battery level of neighbors when selecting the gradient to the sink. However, they do not dynamically change the gradient, even if a node exhausts its energy. Instead, they must wait until the subsequent sink flooding to update the battery level and the route. A similar approach is described in [1], where each node knows the "heights" of its neighbors (number of hops to the sink). If the battery level of some node drops, it increases its height and propagates this new information to its neighbors. A similar idea of increasing the routing costs through a node with a low battery level is used for geographic routing in *GEAR* [8].

The clustering protocol in [10] is an extension of LEACH and falls into the second category as it uses full topology and battery information at the base station to compute the best cluster heads. Nevertheless, centralized clustering approaches are often costly and cannot adapt to rapid topology or network changes.

A number of reinforcement learning based routing protocols have emerged in recent years. Most (including FROMS) are based on or inspired by Q-Routing [4], which uses Q-Learning to find the minimum delay route between a sink and a source in a wired network. Similar approaches use minimum-delay or maximum aggregation rate as cost metrics and are summarized in [2]. The main advantage of machine learning based routing protocols is their ability to learn global values by only exchanging local information.

To evaluate the behavior of our protocol, we adopted the evaluation metrics presented in [1], namely considering first node death for network lifetime and the remaining energy histogram for evaluating energy expenditure spreading. Our route cost functions, instead, are inspired by the analogous functions in the LMMER extension of Directed Diffusion [7] which computes the overall route cost given the number of hops and the remaining energy. More information about our evaluation techniques and metrics wil be given in Section V.

III. ROUTING FRAMEWORK

We carry out this study of the behavior of energy-based cost functions by introducing them into the machine learning framework of FROMS [5], our routing protocol for multi-source, multi-sink data delivery. Its key feature is the application of Q-learning, a reinforcement learning technique, to identify the best data routes in terms of a user-defined cost function. The protocol uses small amounts of information exchanged among neighbors, called feedback, to evaluate the cost of using a particular neighbor as the next hop for data on the path to one or more sinks. This data is piggybacked on the usual DATA packets [12], and thus does not incur additional overhead.

A. FROMS overview

Figure 1 summarizes the key actions of FROMS. Fundamentally packets flow from the sinks to the sources to announce their data requirements (SINK_ANNOUNCE) and from sources to sinks to deliver data (DATA).

```
1 : init:
2 :    init_cost_function();

3 : on receive(SINK_ANNOUNCE):
4 :    registerRoute(sinkID, neiID, hops,
            neiBattery, dataType)

5 : on receive(DATA):
6 :    registerFeedback(DATA.feedback)
7 :    // route packet to next hop(s)
8 :    if (!routes)
9 :       init_routes();
10:       estimate_routes(cost_function);
11:    route = select_route(routes);
12:    DATA.routing = route;
13:    DATA.feedback = best_route_cost;
14:    broadcast(DATA);
```

Fig. 1. The main FROMS algorithm

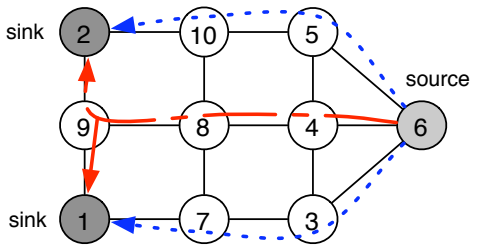

Fig. 2. The topology used in our experiments with two main routes to the sinks.

Initialization: Sink announcements are used to fill the routing tables on each node. The FROMS routing tables are slightly different from traditional tables, as they maintain information on all possible routes to the sinks, not only the best. For example, for the sample topology of Figure 2, the routing table at node 6 will contain three entries to reach sink 2: one through node 5 with cost 3, one through node 4 with cost 4, and another through node 3 with cost 5. For the purpose of this paper, we also store the remaining battery level of each neighbor (lines 3-4).

Routing: When data arrives, it contains a set of destination sinks, and FROMS must decide which neighboring node(s) to use as the next hop(s) toward each sink (line 7). If this is the first time the node has forwarded packets, the routing options are initialized (line 9) and their cost is estimated (line 10). This cost, called the *Q-value* is based on the routing table entries filled with information from the SINK_ANNOUNCE, and is typically a reasonable estimate of the real route cost. Q-values are updated during the network lifetime to reflect learned information. Note that because data packets must reach multiple destinations, routes could be very complex. For example, in Figure 2, the source node must consider sending the data for both sinks through 5, 4, or 3 as the next hop, or splitting the packets for each sink, e.g.,

sending data through node 5 to reach sink 2 and through node 3 to reach sink 1. Additional details on how we manage this complexity are available in [5].

Once routes have been initialized, FROMS selects a route (line 11) either to *exploit* the best known routes or to *explore* routes with non-optimal costs. The choice is based on an *exploration strategy* that balances the often higher cost of exploration with the probability of identifying a lower, best cost route. Before broadcasting the data packet to the selected next hop(s) (line 14), FROMS attaches the best available Q-value as a feedback value (line 13).

Feedback: This piggybacked feedback represents the key to learning in FROMS. Since all nodes overhear packets from their neighbors, we assume each extracts the feedback from them. This is then used to update the local Q-values for the data packet source (line 6).

These three procedures, (1) sink announcements and initial route cost estimates, (2) route selection to explore/exploit, and (3) feedback delivery to update costs, function together to learn the real costs of all routes in the network.

Parameters: FROMS is controlled by two key parameters: a learning parameter $\gamma = [0, 1]$ and an exploration parameter $\epsilon = [0, 1]$. The first, γ, determines how the Q-values are updated when feedback is received. Roughly, with $\gamma \approx 0$, learning is extremely slow and many feedback values are required to learn the actual route costs. However, Q-values change very little with each feedback, meaning that the system behavior changes gradually and is more predictable. Instead, with $\gamma \approx 1$, learning is very fast and old Q-values are essentially ignored and replaced directly by the feedback values. With a static, hop-based cost function in FROMS [5], we showed that a learning rate of 1 is the most appropriate to quickly learn route costs.

The second parameter, ϵ, controls the exploration strategy of FROMS, defining the exploration/exploitation ratio. For example, with $\epsilon = 0.7$, the best route will be chosen with probability of 0.7 (exploitation) and a random route will be chosen with probability $1 - 0.7 = 0.3$ (exploration).

Cost function: The cost function is used both to initialize the Q-values with route estimates and to update the Q-values during the learning process. Additional details on cost functions and how to design them to meet certain optimization goals are given in Section IV.

B. Key Properties of FROMS

There are some key FROMS properties worth mentioning. In terms of implementation, the protocol requires memory sufficient to store all possible routes. The size grows with the number of possible combinations of next hops to reach multiple destinations. We have evaluated several straightforward, effective heuristics that store only the most promising routes. These allow the memory footprint to be reduced to meet hardware limitations. Next, the processing requirements for selecting routes and updating Q-values is negligible. More significant processing is needed for route initialization, however this takes place only infrequently when a new sink joins.

In terms of optimality, in a perfect environment with reliable communication, FROMS finds the optimal route from a source to multiple destinations. In an unreliable environment, such as an error-prone WSN, it finds semi-optimal routes through exploration and learning, thus reducing energy expenditure in terms of the number of broadcasts to deliver a data packet (see [5]). Later in this paper we show how to use FROMS not only to reduce the number of hops to reach multiple sinks in a network, but also to spread the energy expenditure during the network lifetime.

The most important property of FROMS and of any Q-learning based routing approach is its adaptability to node failures and mobility. These are, in fact, topology changes and new route costs can be quickly re-learned by exploration and Q-value updating. We exploit exactly this property of FROMS in this paper, where the remaining battery level on the nodes changes quickly and thus requires quick adaptation by the routing protocol.

IV. DESIGNING ROUTING COST FUNCTIONS

FROMS was originally developed and tested with a cost function based only on hop count and the optimization objective to minimize the number of broadcasts per packet to reach all destinations. This cost function takes advantage of the broadcast nature of the wireless communication medium, and assumes that a single broadcast transmission is sufficient to send a packet from a node to multiple of its neighbors. However, as FROMS has only a localized view of the network, it can only estimate the real route cost. For each possible route it computes this estimate, called the initial Q-value, based on the information in the routing table obtained from the sink announcements.

Given that each routing decision in FROMS consists of possibly several next hops, e.g., through both nodes 3 and 5 in Figure 2, the initial estimate, E_{hops}, for a node to route to some subset of all destinations $D_i \subseteq D$ through a single neighbor, n_i, is computed as follows:

$$E_{hops}(n_i) = \left(\sum_{d \in D_i} hops_d^{n_i} \right) - 2(\mid D_i \mid -1)$$

where neighbor n_i has an estimate of $hops_d$ hops to each of the sinks $d \in D_i$. To calculate the full cost to route to all required destinations through possibly multiple neighbors, the full estimated cost is:

$$E_{hops}(route) = \left(\sum_{i=1}^{k} E_{hops}(n_i) \right) - (k - 1)$$

where k is the number of neighbors selected to serve as the next hop to reach all destinations.

Both formulas compute the sum of the number of hops over all included neighbors to reach all destined sinks, taking into account the broadcast advantage of the current hop *and* the next hops. When using a hop-based cost function for FROMS, the Q-value for each route is equivalent to the estimated hop count:

$$Q_{hops}(route) = E_{hops}(route)$$

In this paper, however, our goal is to study the behavior of energy based cost functions. For this, we combine the hop-based cost estimate with battery level information. Specifically, to best spread the energy consumption throughout the network, we consider the minimum battery level of all nodes on the route $E_{battery}(route) = \min_{n_i \in route} battery$. In terms of Q-values, we still keep a single Q-value per route, however it is based on two components, the estimated hop count cost and the estimated minimum battery level:

$$Q_{comb}(route) = f(E_{hops}, E_{battery})$$

The function f that combines the two estimates into a single Q-value is based on a simple and widely used function, as in:

$$f(E_{hops}, E_{battery}) = hcm(E_{battery}) * E_{hops}$$

hcm is the hop-count-multiplier, a function that weights the hop count estimate based on the remaining energy. For simplicity we drop the "estimation" and denote the Q-value components as $hops$ and $battery$.

The following shows how to adapt FROMS to this new cost function and how to design it to meet certain optimization goals in the network.

A. FROMS adaptation to energy-based cost functions

FROMS is designed to accept any cost function that uses information available in the initially constructed routing tables (see Section III). Because our cost function is based on two elements, hop count and battery, rather than the original single element, slight modifications are required to FROMS:

- the feedback section of the DATA packet is extended to carry both components of the Q-value
- the stored Q-value for each route is extended to store both its components

Note that although for all routes we keep *both* components of its Q-value, they are always considered together. In other words, the feedback represents the battery level and hop count for a single routing option, not the battery level for one route and the hop count for another. Similarly, learning is done over both values together, not individually. The separation of the Q-value into two components is mainly used for evaluation and administrative purposes.

B. Designing energy-based cost functions

We next show how to design an energy-based cost function to meet two different optimization goals:

1) give preference to minimizing cost per packet, while trying to spread the energy expenditure
2) give preference to energy spreading, while trying to minimize the cost per packet

The key is the definition of the function discussed previously $f(hops, battery) = hcm(battery) * hops$, which multiplies the hop count of a route by a hop-count multiplier (hcm), which is itself a function of the battery meeting the optimization goal. The purpose of hcm is to increase the cost of some route when the nodes on it have depleted their batteries, making that route *less appealing* in comparison to routes composed of nodes with better battery levels.

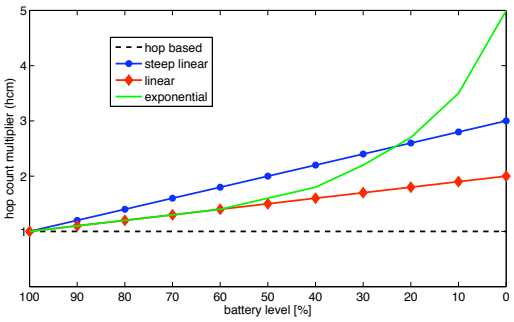

Fig. 3. Hop count multiplier (hcm) function for different optimization goals.

Figure 3 shows four different hcm functions. If the battery level is completely irrelevant, then $hcm(battery)$ is a constant and $f(hops, battery)$ is reduced to a hop-based function only. Instead, if the desired behavior is to linearly increase f as the battery levels decrease, a linear hcm function should be considered. Figure 3 shows two linear functions. The first (labeled linear), has minimal effect on the routing behavior. For example, a greedy protocol which always uses the best (lowest) Q-values available, when faced with the following two routes with $f(1, 10\%) = 1.9$ and $f(2, 100\%) = 2$, will select the shorter route even though the battery is nearly exhausted. Even when faced with longer routes of length 2 and 3 respectively, it will use the shorter route until its battery drops to 40%. When their values become $f(2, 40\%) = 3.2$ and $f(3, 100\%) = 3$, the protocol will switch to the longer route. Thus, this trade-off of weighing the hop count of routes (their length) versus the remaining batteries must be taken into account when defining hcm. In our case a steeper linear function meets the first optimization goal.

The main drawback of linear hcm functions is that they do not differentiate between battery levels in the low and high power domain. For example, a difference of 10% battery looks the same for $20 - 30\%$ and for $80 - 90\%$. Thus, to meet the second goal we require an exponential function that starts by slowly increasing the value of hcm with decreasing battery, initially giving preference to shorter routes. However, as batteries start to deplete, it should more quickly increase hcm in order to use other available routes, even if they are much longer, thus maximizing the lifetime of individual nodes. Of course, such a function that gives preference to longer energy-rich routes will also increase the per packet costs in the network.

V. SIMULATION AND EVALUATION

To test our hypothesis about the behavior of the energy-based cost functions discussed in Section IV, we conducted several simulations in Matlab, assuming reliable communication. For test purposes we used the network topology given in Figure 2, as it illustrates an interesting case for routing to two different sinks, in which different routes with different costs exist and where the energy expenditure can be spread to

TABLE I
EVALUATION METRICS.

Metric	Units	Desired
First node death	pkts	high
Network disconnection	pkts	high
Living cost per packet	transmissions	low
Full cost per packet	transmissions	low
Living energy histogram	percent remaining battery	one compact cluster

TABLE II
GREEDY ROUTING STATISTICS FOR DIFFERENT COST FUNCTIONS.

Cost function	Living cost per pkt	Full cost per pkt	First node death	Network discon.
Hop-based	5 pkts	4.5 pkts	100 pkts	200 pkts
Linear	4.33 pkts	4.5 pkts	150 pkts	200 pkts
Linear steep	4.38 pkts	4.5 pkts	160 pkts	200 pkts
Exponential	4.44 pkts	4.5 pkts	180 pkts	200 pkts

prolong network lifetime.

To measure the energy balancing features, we assigned each node a "battery budget" of 100 broadcasts, except for the source which has an infinite budget. This is required, because if the source has limited energy, it will be the first network node to die and no other behavior can be evaluated.

We show the behavior of FROMS for two different parameter sets, which we call *greedy routing* and *proactive learning*. The first always selects the best available route(s) ($\epsilon = 0, \gamma = 1$), yielding a simple strategy with well-defined non-probabilistic behavior. It is therefore easy to evaluate and very well suited for initial comparisons of the cost functions. The second uses various exploration rates and a learning rate $\gamma < 1$, which smooths possible fluctuations of the learned values. We determined experimentally that $\gamma = 0.7$ provides a good trade-off between smooth Q-value updates and fast learning.

A. Cost functions

As explained in Section IV we study the behavior of four different routing cost functions, each based on a different hcm function shown in Figure 3, and each designed to meet a specific optimization goal.

B. Evaluation metrics

To evaluate the behavior and performance of the cost functions we adopt the metrics summarized in Table I. In terms of network lifetime, *first node death* is the most important. Even in case data delivery can continue after the first node fails, there is no guarantee that if the sources or sinks change, e.g., due to mobility or random node failures, the network will continue to function due to possible network disconnection. Nevertheless, for comparison purposes, we provide also the full network lifetime (*network disconnection*), which identifies the point at which data no longer reaches all the sinks.

In terms of routing costs we consider the *living cost per packet*, which denotes the costs per packet until the fist node in the network dies and *full cost per packet*, which is the accumulated cost per packet delivered to all sinks during the full simulation run (until network disconnection). Both are given in terms of the number of transmissions needed to reach both sinks. Since we do not employ a low-level retransmission scheme for lost packets, this metric is the same as ETX (Expected Number of Transmissions).

The last but very important metric is the *energy spreading histogram* at first node death. We adopted this from [1], where the desired scenario is that at any point during the lifetime of the network, the energy spreading histogram should exhibit a compact cluster, with all nodes having more or less the same remaining energy. We consider the energy spreading histogram at only one point in time, namely first node death, and evaluate the energy spreading yielded by the routing cost function.

C. Greedy routing

In greedy routing, we fix the learning rate of the algorithm to 1 and the exploration rate to 0, thus forcing nodes to use only the best available routes, those with the lowest Q-values. Here we conducted several experiments with the different cost functions, but ran the simulation exactly once with each parameter setting, since there is no probabilistic behavior in greedy routing.

Figure 4 shows the energy spreading histograms at first node death for each of the cost functions. As expected, the exponential cost function performs best and builds a compact cluster around 10% remaining energy. The other two energy-based functions build more scattered clusters and the hop-based function quickly depletes the energy of the nodes on the best route, leaving other nodes completely unused.

Table II summarizes the results of greedy routing in terms of first node death, network disconnection and costs per packet for the different cost functions. Again, as expected, the exponential energy-based cost nearly doubles the network lifetime until first node death, while the hop-based one completely depletes the nodes on the best route very quickly. The network disconnection is the same for all cost functions because the FROMS routing protocol switches automatically to other available routes after node failure. For the same reason the full costs per packet are also the same for all cost functions.

Table II also illustrates the exploratory nature of energy-based cost functions. In the original FROMS algorithm the greedy routing strategy enforces the use of only the best available routes. Thus, other routes with unfavorable cost estimates will never be used and their real costs will never be learned. However, when using the energy-based cost functions whose values change over time, even the greedy approach applied in FROMS switches between routes with different hop costs, because it depletes their energy thus changing their Q-values. At the same time the protocol is able to learn also their real hop-based costs. In other words, even if the exploration rate is set to 0, FROMS is *implicitly* exploring the network by using the dynamic cost function. The real power of this property will be shown in the next section, where we use proactive learning.

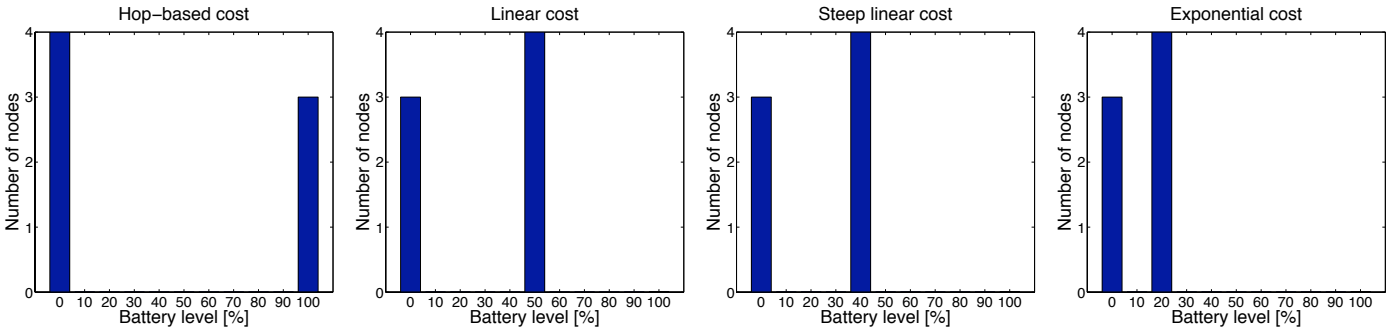

Fig. 4. Energy histograms of nodes at first node death. Sinks and source are not included.

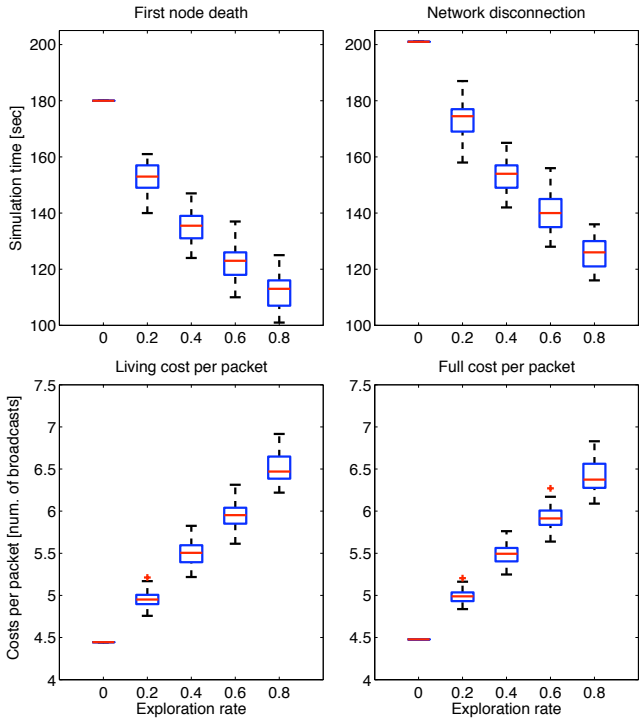

Fig. 5. Exponential energy-based routing cost function with learning rate of 0.7 and different exploration rates. The box has lines at the lower quartile, median, and upper quartile values. Whiskers extend from each end of the box to the adjacent values in the data within 1.5 times the interquartile range from the ends of the box. Outliers are displayed with a + sign

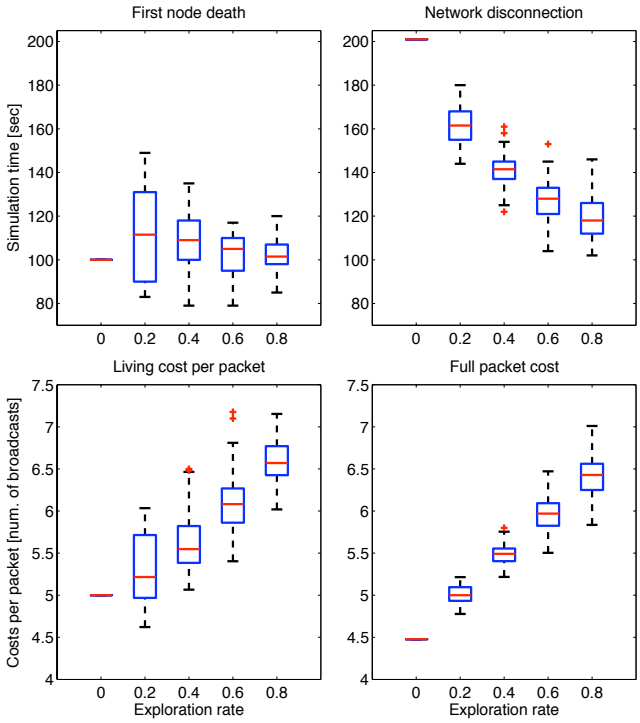

Fig. 6. Hop-based routing cost function with learning rate of 0.7 and varying exploration rates. Figure 5 explains the graph format.

D. Routing with proactive learning

Next we consider FROMS with learning enabled, with $\gamma = 0.7$, and vary the exploration rate.

We see again that the implicit exploratory behavior of energy-based cost functions makes explicit exploration unnecessary. With proactive learning, FROMS uses non-optimal routes with some probability $1 - \epsilon$ and learns their real hop-based and battery-based costs. On the other hand, it also benefits from the *implicit* exploration provided by the dynamic energy-based function. As we can see from Figure 5, FROMS without exploration (exploration rate $\epsilon = 0$) achieves the best results in terms of all evaluation metrics. With exploration rate $\epsilon > 0$, all evaluation metrics drop because of the double exploration behavior. Thus, explicit exploration is clearly not desirable in such an environment.

Figure 6 shows the results from the hop-based cost function with a learning rate of 0.7 and varying exploration rates. First we can see that the overall results compared to the exponential energy-based function are worse, exactly as predicted in Section IV. Second, we see that exploration does make sense here, since the results actually improve through exploration. For example, the time to first node death is prolonged from 100 to 110 packets (Figure 6 top left) because exploration also uses non-optimal routes, thus spreading the load of the nodes. With higher exploration rates, the protocol uses more non-optimal routes, increasing the per-packet costs.

VI. Discussion

Next we offer a brief summary of the combined results from the theoretical analysis in Section IV and the experimental

evaluation in Section V.

First, we have shown that an energy-based cost function can achieve reasonable energy consumption balance in a WSN when used within a reinforcement learning based routing protocol like FROMS. Variants of these cost functions achieve different optimization goals in the network and must be carefully designed, as outlined in Section IV.

The more important result, however, is the outcome of the analysis of the behavior of FROMS with these dynamic cost functions. Fundamentally, we observed that such a reinforcement learning based routing approach requires exploration to discover new lower-cost routes in the network, however, this exploration can be done either in the traditional way with explicit exploration or in a new way by using a dynamic cost function whose evaluation changes over time. Such cost functions force the learning algorithm to switch between available routes, thus also learning the real hop costs of initially unfavorable routes. It should be noted that this behavior is also partly due to our good initial estimation of routes. If this were not the case the routes selected at the beginning of the network lifetime would be highly unfavorable, negatively affecting network lifetime.

This is a new result in the sensor network environment, which stands somewhat in contrast to the expectations of applying reinforcement learning to routing. It is widely accepted that in general such algorithms are costly because of the explicit exploration; that they are slow and uncertain to converge, again due to exploration; that dynamic cost functions will further slower them down and make them even more costly; and that they are complicated to implement because of the many parameters for exploration/exploitation etc.

Here, instead, we clearly demonstrated that such a scheme not only works well with dynamic cost functions, but that it actually makes the explicit, possibly costly, exploration redundant. Thus reinforcement learning based protocols actually become easier to implement, convergence is no longer relevant and their behavior is flexible to dynamic scenarios.

VII. CONCLUSION AND FUTURE WORK

The contribution of this work is two-fold: first, it presents an extension of our machine learning based protocol to use a dynamic energy-based cost function, thus effectively balancing energy consumption in the network over time. Second, our study of the behavior of this protocol reveals that the explicit exploration found in traditional reinforcement learning approaches becomes redundant due to the implicit learning caused by the dynamic cost function. This is a particularly important result, as the exploration process can be costly, wasting energy in the resource constrained environment of WSNs. Here, instead, we showed that even with a greedy exploitation strategy, the implicit exploration is sufficient to effectively use the system resources.

This work represents only the beginning of our energy-based study. Our next step is to move into a more realistic simulation environment, namely Omnet++, in order to fully test the behavior of the energy-based functions for FROMS. Specifically, we will consider multiple, larger, random network topologies, increase the traffic loads by varying the number of sources, and introduce sink mobility and node failure. We will also include more sophisticated energy consumption models, including message reception cost. Further work will involve the evaluation of FROMS on top of realistic, energy-efficient MAC protocols employing sleep cycles, as these will challenge the feedback process of FROMS and thus its learning. Additionally, we will continue our theoretical study to analyze machine learning algorithms for the general wireless ad hoc scenario, developing protocols which are even more flexible, cost-effective and stable than existing ones.

ACKNOWLEDGMENT

The work described in this paper is supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

REFERENCES

[1] C. Schurgers and M. B. Srivastava, "Energy efficient routing in wireless sensor networks," in *Proceedings of the IEEE Military Communications Conference*, vol. 1, 2001, pp. 357–361.

[2] A. Förster, "Machine Learning Techniques Applied to Wireless Ad-Hoc Networks: Guide and Survey," in *Proceedings of the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 2007.

[3] P. Beyens, M. Peeters, K. Steenhaut, and A. Nowe, "Routing with Compression in Wireless Sensor Networks: A Q-Learning approach," in *Proceedings of the 5th European Workshop on Adaptive Agents and Multi-Agent Systems (AAMAS)*, 2005.

[4] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," *Advances in Neural Information Processing Systems*, vol. 6, 1994.

[5] A. Förster and A. L. Murphy, "FROMS: Feedback Routing for Optimizing Multiple Sinks in WSN with Reinforcement Learning," in *Proceedings of the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 2007.

[6] Z. Li and H. Shi, "Design of gradient and node remaining energy constrained directed diffusion routing for wsn," in *Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing*, 2007, pp. 2600–2603.

[7] A. Bachir and D. Barthel, "Localized max-min remaining energy routing for wsn using delay control," in *Proceedings of IEEE International Conference on Communications*, vol. 5, 2005, pp. 3302–3306.

[8] Y. Yu, R. Govindan, and D. Estrin, "Geographical and Energy Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks," UCLA Computer Science Department, Tech. Rep. UCLA/CSD-TR-01-0023, 2001.

[9] I. Stojmenovic and X. Lin, "Power-aware localized routing in wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 12, no. 11, pp. 1122–1133, 2001.

[10] K. Y. Jang, K. T. Kim, and H. Y. Youn, "An energy efficient routing scheme for wireless sensor networks," in *Proceedings of the International Conference on Computational Science and its Applications*, 2007, pp. 399–404.

[11] F. Silva, J. Heidemann, R. Govindan, and D. Estrin, *Frontiers in Distributed Sensor Networks*. CRC Press, Inc., 2003, ch. Directed Diffusion.

[12] A. Egorova-Förster and A. L. Murphy, "Exploiting Reinforcement Learning for Multiple Sink Routing in WSNs," in *Proceedings of the 4th IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (MASS)*, 2007.