A Feedback-Enhanced Learning Approach for Routing in WSN

Anna Egorova-Förster[1] and Amy L. Murphy[1,2]

[1] University of Lugano, Switzerland,
{anna.egorova.foerster, amy.murphy}@lu.unisi.ch
[2] IRST-ITC, Trento, Italy

Abstract. A new modality for sensor networks emerges when considering multiple, distributed base stations that collect data from sensors. This scenario reverses the typical multiple-source, single-sink scenario, and requires new techniques to efficiently send data from single-sources to multiple-sinks. While an offline approach with full topology information can build the optimal data forwarding tree, the challenge we address here is to optimize data forwarding with only information exchanged among one-hop neighbors. The novelty of our approach lies in the use of an iterative *learning* technique that explores alternative routes by locally sharing *feedback* regarding route fitness. This paper presents our approach as well as an evaluation showing that the learned paths lead to increases in network lifetime of up to 50% over an approach without learning.

1 Introduction

Technological advancements in hardware and wireless networking have recently raised the potential for wireless sensor networks (WSN) with much effort concentrating on collecting vast amounts of sensed information at a single, powerful base station. Although a variety of applications can exploit this setup, the network algorithms developed for it have limited use in sensor-actuator networks that exploit data *inside* the network. In such scenarios, actuator devices are physically dispersed beside the sensors where they collect and act on sensor data. Exploiting a traditional, centralized approach requires collecting information at a base station and re-sending it to the actuators. This, however, unnecessarily increases both the amount of data transmitted and the latency between data collection and actuation.

By taking this sensor/actuator scenario as our motivation, our work essentially reverses typical WSN assumptions and instead of sending multiple pieces of data to a single sink, we target sending a single piece of data to multiple sinks. Our work trivially extends to support multiple data sources. Within this context, our goal is to minimize the overall network cost, thus increasing its lifetime.

Recent research efforts have resulted in a wealth of routing protocols to discover energy-efficient paths between data sources and destinations. While they are effective in connecting two points, the minimal tree connecting a source

and multiple sinks may not contain any of these pairwise routes. Our challenge, therefore, is to discover this minimal tree of routes without requiring global information about the network topology.

Our approach is an in-network protocol that incrementally *learns* about good paths in the network. To achieve this, we assume that sink nodes advertise their data needs with a simple broadcast protocol, establishing basic, sub-optimal routing information. To find better paths, neighbors exchange information about the fitness of various routes, using this information to incrementally improve forwarding decisions and overall decrease the network cost for the data to reach all sinks. During this process, each node actually learns multiple, equal cost routes, which can be cycled through, thus further increasing the lifetime by spreading the routing load across more nodes.

This paper continues in Section 2 presenting our routing protocol. Section 3 provides a discussion and simulation results demonstrating the effectiveness of our approach. We then outline related work before Section 5 concludes the paper with our future plans.

2 Approach

The goal of our protocol is to find the shortest possible path for data to follow from its source to all interested sinks. One possible path, actually a tree, is formed by the union of the individual paths from the source to each sink, however a shorter path often exists. While the tree discovered by our approach may not be the theoretical optimum, it is likely to be an improvement over the individual paths. The challenge is to identify this tree without full topology information and without exchanging global information. The main task of our protocol is to update local information regarding "next-hops" to reach all sinks such that the resulting forwarding tree is as small as possible.

We accomplish this with a *three phase protocol*. The first establishes basic information at each node regarding the identity of the sinks and initial path information to each sink. The second phase sends sensor data, explores routes, and learns the accurate shared costs of these routes. The third phase is the steady-state, in which sensor data packets are routed along the best available routes without exploration. The sequence of phases is not strict, meaning that if a new sink requests data, or if a new, shorter path is introduced, the system will asynchronously return to the first and/or second phases.

It is worth noting that our routing protocol tolerates both node failure and loss of data and feedback packets. Also, while the protocol does not prevent routing loops during the exploration phase, it still ensures that the packet eventually reaches the destination, unless it is corrupted during transmission.

2.1 Phase I - Sink announcement

The first phase of our protocol requires each sink node to broadcast its request to the entire network. As this packet propagates, nodes receive information about

DATA REQUEST	
sinkID [*int*]	*A*
time_stamp [*sec*]	78
coordinates $[x, y, r]$	$[100, 45, 8]$
expire_after [*sec*]	1000
TTL [*int*]	10
hops [*int*]	1

DATA PACKET				
Data:	data	...		
	sourceID [*int*]	20		
	time_stamp [*sec*]	78		
Feedback:	forID	34		
	RLE [*int*]	12		
Routing:	neighborID	45	12	...
	sinks(*array*)	(B, C)	(A, D)	...
	max_cost	4	9	...

Fig. 1. Data request and data packet formats.

the cost along available paths to the sink nodes. The request message outlined in Figure 1 is fairly straightforward. The most critical info it carries includes the identifier of the sink node and the packet time to live (TTL), indicating the limit of the dissemination of the request message. The hops entry, initialized to 1 at the sink, increments each time the request is forwarded.

When a node receives a request message, it consults its routing table, shown in Figure 2 for the arbitrary node with identifier 20. If an entry already exists in the table for this sink/neighbor combination with a smaller number of hops traveled, the message is not processed. Otherwise, a new entry is added or an existing one is updated and the message is re-broadcast with updated information.

2.2 Phase II - Exploration

Phase II represents the core of our protocol as it is during this phase that the best route from the source to all sinks is discovered. The key points to our approach are the *fitness function* that estimates the effectiveness of the route, the data structures used to make routing decisions, and the *exploration strategies* to determine which routes are taken.

To reduce data sending costs, all packets are sent in one-hop broadcast mode, not unicast, to all neighbors. Therefore, if a packet should be processed by more than one neighbor, a single packet is sent with information inside specifying which nodes should receive it. In Figure 1, this information appears in the *Routing* section of the data packet.

Phase II initiates when a data packet is received at a node and it must make a decision concerning how to route that packet. Each data packet carries *Routing* information concerning which sinks it is expected to reach by going through which nodes. For example, in Figure 1, the data packet has to reach sinks (B, C) by going through node 45 and sinks (A, D) through node 12.

Fitness Function (Route Length Estimate). Given that the goal of our protocol is to find the *best possible* route to all sink nodes, we must define precisely the metric for evaluating routes. By studying the environment, several properties emerge. For example, remaining battery power at the nodes or link quality between nodes can affect the quality of the path. In general, the fitness function

Neighbor ID	45			54	12	
Route to Sink ID	A	B	C	B	A	D
#Hops	1	3	2	2	6	4

Fig. 2. Routing information maintained by sensor node 20 with path lengths to all known sinks obtained in Phase I.

can be calculated based on multiple variables related to the appropriateness of the path for routing packets to a particular sink or set of sinks.

In the rest of this paper we use a simple, intuitive fitness function: *Route Length Estimate* (RLE). This metric is the total number of hops that a message is expected to travel to reach all sinks and generally corresponds to the energy requirements to route the packet through the network.

After Phase I, all nodes know locally the cost to send to a given sink through a given node. Based on this information, any node can estimate the cost to send a packet to a *set* of sink nodes through some of its neighbors. However, because the topology is not known, this cost is only an approximation, hence the word *estimate* in our fitness metric. Using Figure 2 as an example, it is known that from node 20, neighbor node 45 can be used to reach sinks (A, B, C) but it is not known whether node 45 has an outgoing shared link to use for these packets, or if it must split the packets to three different neighbors to reach all of the sinks. Therefore, we estimate the worst case remaining cost to send a packet to sinks (A, B, C) through node 45 as the cost to send the packet one shared hop to node 45 and then to split the packet along three different paths after node 45 for the remainder of its route to the sinks. Therefore the cost is $1 + (1 - 1) + (3 - 1) + (2 - 1) = 4$. This equation generalizes to:

$$RLE = \left(\sum_i cost_i \right) - (n - 1) \tag{1}$$

where i is the set of sinks ((A, B, C) for this example), $cost_i$ is the path length to send a packet through the selected neighbor for the i-th sink (1, 3, and 2 above) and n is the number of sinks (3 above).

It is important to emphasize that this is the worst case estimate for the path fitness. It is entirely possible that the packet can continue to share a path after the next hop, in which case the actual cost will be lower. To *learn* a more accurate estimate of the cost, the estimates for the same routes on neighboring nodes are conveyed back to the sending node, in our example from nodes 45 and 12 to node 20. This is done as *feedback* during the routing process, as described later in this section.

Path Sharing Tree. When a node receives a data packet, it ensures that this data packet will reach all of the sinks listed inside the data packet by delegating responsibility for routing the data to one or more of its neighbors. If a packet contains only one sink destination, the routing decision can be taken only by looking at Figure 2, and selecting the entry with the shortest path to that sink.

However, if the data must reach several sinks, the decision is complex. Consider example node 20 in Figure 2. If a data packet must reach all four sinks we have at least the following distinct options: *(i)* 45 for (A, B, C), 12 for (D), *(ii)* 12 for (A, D), 45 for (B, C), or *(iii)* 45 for (A, C), 54 for (B), 12 for (D). This is not a complete enumeration, but it gives only impression of the complexity.

To manage this complexity, we devised a data structure called the *Path Sharing Tree* (PST) whose goal is to organize the available options for selecting the next hops for a data packet to take. Here, we stress only the most important properties of this data structure and, for space reasons, leave out the implementation details which are available in [5].

The data structure must (i) hold all possible route combinations through all neighbors, (ii) update path fitness and (iii) return a desired route for a set of sinks. The implementation of this data structure is important for the overall scalability and performance of the protocol, as the number of combinations grows with large number of neighbors and/or sinks. The PST is pruned to eliminate duplicate routes and to ignore routes with very high costs.

Managing Loops. While the data packet is routed through the network, every node is allowed to explore non-optimal routes, e.g. routes with higher costs than absolutely necessary. This "freedom of choice" makes it possible to learn about better routes in the network, but also opens the risk of routing loops. Our solution allows data to travel in a loop, but prohibits infinite loops. This is accomplished by guaranteeing that the *max_cost* for routing allowed by the data packet always decreases. If this cannot be met by a routing decision, the packet is returned to the sender, who must choose a different path to the sink(s).

Feedback and Learning. While the PST estimates the network cost to all sinks, its initial data is based only on information gathered during Phase I and is therefore the worst-case estimate of the possible path sharing. It is necessary for any downstream nodes to update route costs at its upstream nodes, providing more accurate route length estimates. This information is shared in the form of feedback, piggybacked when the packet is re-broadcast by the next hop. This sending and receiving of the feedback information and the updating of the RLEs throughout the network is our *learning mechanism* that results in improved routing decisions over time.

Consider the example shown in the left of Figure 3 in which sender node 20 forwards a packet to nodes 45 and 12 for sinks (B, C) and (A, D) respectively. When node 45 receives this packet, it must find a path for routing to sinks (B, C) with cost less than 4. It identifies another neighbor, 5, as an appropriate next hop for both sinks, with the cost of 2. Node 45 sets feedback information in the packet that it forwards, indicating that the current estimate of the path known at node 45 is 2. When node 20 receives this feedback, it updates the PST tree, setting the cost for using node 45 as the next hop for (B, C) to the feedback value and updating all other relevant costs and routes.

In general, feedback updates information at the previous hop, however in order to find the *overall* best routes, this information must propagate throughout the network; our mechanism does precisely this. Observe that node 20 now has

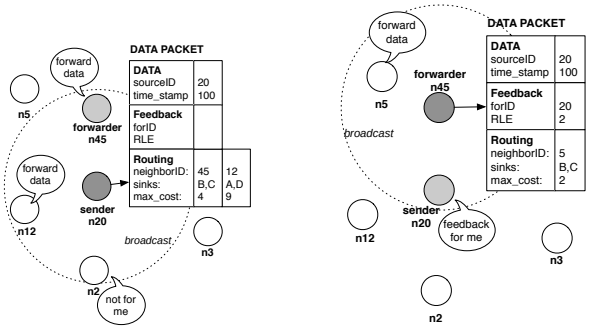


Fig. 3. Sample routing and feedback for a message traveling from node 20 to 45 and from 45 to 5.

a better cost estimate that will affect its perception of the route fitness: it has *learned* a new RLE for this route. Therefore, the next time it sends data along this same route, it will forward the updated cost estimate to its predecessor node. Thus, the information eventually propagates the full reverse path from the sinks to the data senders. This means that for a route to some sink with n hops, the same route must be used exactly $(n-1)$ times in order for all nodes on the path from the source to the sink to have the most accurate information.

Exploration Strategies. While the PST information shows several possible routes and combinations, it is the job of the routing protocol to decide which route to use. We could simply choose the path with the minimum cost. However, it may be that other routes have poor estimates and lower actual costs. Another option is to cycle among all options, to receive feedback from the other nodes to update the RLE. Yet a third option is to assign weights to the options, e.g. larger weights to paths with lower cost. Using these weights, we can probabilistically select a path to use, selecting low cost paths with higher probability. Other route exploration strategies are possible, with the choice depending on application requirements and network properties. The exploration strategy works together with the loop management and ensures that only *allowed* routes are chosen, i.e. their costs are strictly less than the maximum allowed cost.

Every exploration strategy must define the duration of the exploration phase. Again, several options are available based on time, the received feedback, the number of explored routes, etc. In our implementation, the exploration phase ends after a given number of packets fail to trigger an update of the PST. With a constant data rate, this is equivalent to a time-based approach, however it is more flexible for varying data rate applications.

2.3 Phase III - Stable Data Gathering

After the exploration phase terminates, the PST is likely to have several routes with the same, *best*, minimal cost. Alternating among these paths is likely to spread out the load of data forwarding among the available network nodes,

therefore Phase III is characterized by randomly selecting one of the best paths and sending the data packet along it.

It should be noted that if the PST changes for any reason, Phase II is re-initiated. The protocol may exit the stable phase, but assuming the PST eventually stops changing, the system will spend most of its time in Phase III.

3 Discussion and Evaluation

Throughout this section, we examine the behavior of our protocol with two different exploration strategies: randomExplore and noExplore. randomExplore assigns probabilities to each route according to its total cost and uses them when selecting randomly the route to be explored. It stops exploring, when ten consecutive data packets receive no valuable feedback. noExplore uses only the best found routes after Phase I, meaning no exploration of routes with larger costs is ever performed. For comparison we use an implementation of Directed Diffusion's "one phase pull" protocol, directedDiffusion, as described in [7, 9]. We selected this protocol as it is both similar to our approach and has demonstrated good experimental results.

3.1 Discussion

The feedback-enhanced routing protocol presented thus far is based on non-deterministic, probabilistic learning. As discussed in Section 2.2, it guarantees data packets eventually reach the sinks in all phases, however, it does not guarantee to find the optimal route, as this has unrealistic requirements for on-line implementation. Therefore, we apply a standard machine learning technique: randomize and limit the exploration to all heuristically good routes.

The success of the exploration phase is dependent on the topology of the network. With some topologies, e.g. that of Figure 4(a), the optimal route to all sinks is the same as the individual routes to each sink separately, which is the routing information collected in Phase I. In this case, termed *best single routes*, neither randomExplore nor noExplore can find a better route because the best single routes already use the optimal one. Nevertheless, in other cases, benefits in network cost will be found in the exploration phase, as in Figure 4(b) where the gain over the best single routes is 40%.

The topology in Figure 4(c) illustrates an important property of our exploration mechanism: it is *not* guaranteed to find the optimal route. Because we choose randomly among the available routes during exploration, it may happen that the middle route is not explored before the end of the exploration phase. In this case, the optimal route will never be found. This depends clearly on the exploration strategy as it defines how the routes are explored.

Despite the possibility that exploration may not find better routes, it provides two main benefits. First, *most of the time* it does find routes with better fitness through exploration (see the simulation results in the next section). Second, it

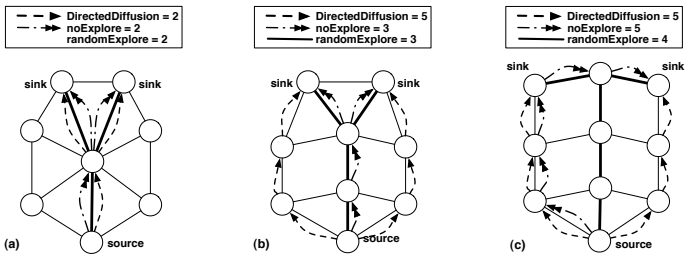


Fig. 4. Various network topologies exhibit key properties of the routing protocol. The gradients for DIRECTEDDIFFUSION and the feedback-enhanced route are shown with arrows and RLE values are shown for DIRECTEDDIFFUSION, NOEXPLORE and RANDOMEXPLORE (after stabilization).

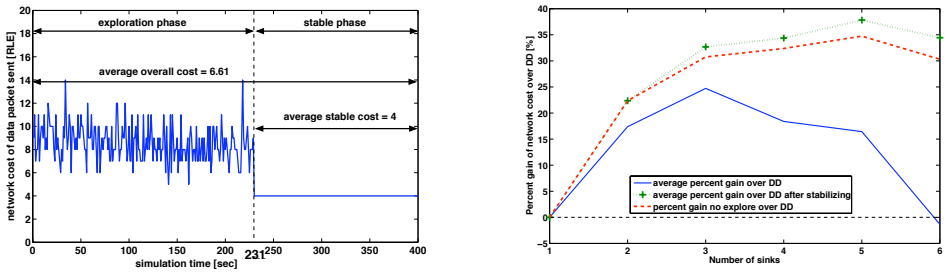
typically finds *many* routes with the same minimum cost, which can be used for managing network lifetime by sharing energy among forwarding nodes.

Extreme scenarios. It is worth abstractly considering some extreme scenarios including both very few and very many sinks. With only one sink, the PST tree will not be built at all because there are no shared routes and the best route to this sink from the routing table will be used directly. The results will be exactly the same as using DIRECTEDDIFFUSION. With a large number of sinks, several properties change. The exploration time increases because more shared routes are possible through a single neighbor. The network cost savings also increase for the same reason. On the negative side, the size of the PST also increases in order to track all possible paths.

Multiple sources and data types. To this point we have only considered a single source in the network. However, when the number of sources changes, neither the protocol nor the properties of the exploration change. The feedback relates to the sinks and not the sources or the data, therefore it remains unchanged. Finally, only one PST is needed per node as this PST holds all possible routes. The same holds for multiple data types.

Repair after failure. Our protocol is also tolerant to connection failures. If some neighbor is no longer reachable, all corresponding routes are deleted from the PST of the node that discovered the failure and the next best routes will be used. If the cost of the new route is larger than before, this information is passed as feedback to the neighbors, updating their PSTs, triggering a new exploration phase, and causing new routes to be explored and used.

Bandwidth requirements. We have already shown that the protocol does not increase network costs in the stable phase since feedback information is piggybacked on normal data packets. Although the exploration phase increases network costs due to the exploration of non-optimal routes, this temporary increase is compensated for in the stable phase, as seen in the simulation results.



(a) (b)

Fig. 5. (a) The network routing costs for data from a single source to all sinks in a sample, random topology with 6 sinks. The two phases of RANDOMEXPLORE are shown: exploration until time 231 followed by stable. (b) Percent gain over DIRECTEDDIFFUSION in network costs for RANDOMEXPLORE after stabilizing, RANDOMEXPLORE total and NOEXPLORE.

3.2 Simulation setup and results

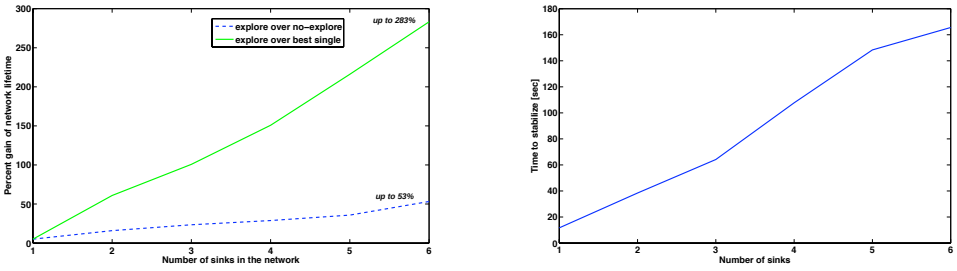
Our simulation study was conducted using OMNeT++ v3.2 and the Mobility Framework v1.06 (http://mobility-fw.sourceforge.net/). Both results and simulation code are available on our website (http://www.inf.unisi.ch/projects/mics).

Our evaluation compares two versions of our feedback-enhanced protocols NOEXPLORE and RANDOMEXPLORE against an implementation of the "one phase pull" DIRECTEDDIFFUSION. All the experiments reported here consist of 500 runs over 125 different random topologies of 50 nodes with one single source in the network. Each topology is guaranteed to be connected and is fixed at the beginning of the simulation. To simulate lifetime, we assign each node an initial energy quota and decrement this value when transmitting.

Before showing any comparisons, Figure 5(a) intuitively shows the behavior of the protocol for a single run. Before stabilizing, the network costs are irregular because packets are sent through different routes with varying costs. Once the stable phase begins, only the best routes are used.

Based on these observations, we define two metrics for comparison: *overall cost* and *stable cost*. The first denotes the average network cost during the whole run (both phases), 6.61 in this case. The stable cost averages only the cost during the stable phase, 4 in this case. While clearly the additional costs during the exploration phase are important as they are part of the total energy expended by the system, this value becomes less relevant with long simulation runs. The second metric, on the other hand, shows clearly the gains that our approach can achieve after short, shared routes are identified.

Figure 5(b) shows the improvements of our protocols compared to DIRECTEDDIFFUSION, reporting the percentage in performance gain on the same topology for each protocol. This metric eliminates differences among random topologies, such as the actual length (number of hops) of the paths between source and sinks.



(a) (b)

Fig. 6. (a) Network lifetime of our protocols compared to DIRECTEDDIFFUSION. (b) Duration of the exploration phase for RANDOMEXPLORE.

The stable cost of our protocol achieves significant gains, up to 35% for 5 sinks. Much of the gain comes simply by exploiting the PST to find shared routes but in most cases, additional gains are achieved by the exploration. However, when considering the overall cost, our gains are not as significant over DIRECTEDDIFFUSION. In fact, the last data point for the overall cost with 6 sinks is below that of DIRECTEDDIFFUSION. Because the overall cost depends heavily on the length of the simulation run, over time these additional costs are amortized by the gains achieved with shorter routes discovered during exploration.

At the measurement for two sinks, RANDOMEXPLORE is unlikely to find a better route than NOEXPLORE because of the limited number of alternate routes that exist. As the number of sinks increases and the number of available routes correspondingly increases, the potential improvements due to exploration grow. Also the slight decrease in the trend at the last data point is most likely due to the random behavior of our protocol and noise in the network and we do not expect it to continue.

Next we consider the potential gains in network lifetime by exploiting multiple identical cost links. Again our hypothesis holds: alternating among the equal cost paths increases network lifetime, as measured by the time to the first node failure. Figure 6(a) shows that when comparing the stable phase after RANDOMEXPLORE against DIRECTEDDIFFUSION, our approach can achieve up to 283% longer lifetime than DIRECTEDDIFFUSION. Compared to NOEXPLORE, our approach still shows up to 53% improvement, demonstrating that learning and exploration can achieve significant benefits.

We also note the average time to stabilization for various numbers of sink nodes, as this affects the overall cost. Figure 6(b) shows an increase in stabilization time with more sinks because it takes more time to receive all the feedback. Nonetheless, considering that many real applications gather data for weeks or months, such an increase in exploration time is acceptable.

In addition to the presented results, we ran simulations with dense topologies, increasing connectivity among the nodes. This increase implies two trends: first,

routes to the sinks are shorter reducing the exploration phase; and second, there are more neighbors, implying the number of routes available to explore increases. As confirmed by simulations, these two changes compensate for one another, yielding no measurable difference for any experiments.

Additionally, we gathered information about the data delivery rates at the sinks, for setups with 1 to 6 sinks. We achieved results varying from 97% to 99%, which we expected since the data rate in the network is relatively low (once per second). With increasing data rates, we expect it to decrease.

4 Related Work

In this section, we compare our approach to content based routing in sensor networks and the application of learning techniques in networking. The presentation is not exhaustive, but is intended to put our work in context.

One way to view our approach is as an implementation of content-based networking [3], a routing framework where data is sent from the source to the destinations based on data interests expressed by the destinations. Such an approach is relevant for sensor networks as they are data driven as opposed to address driven [6]. A well known instantiation of content-based networking for sensor networks is Directed Diffusion [7, 9] where routes from the source to the destinations are established on-demand based on interests flooded through the network. The best paths from the sources to the sinks are reinforced as data flows through the network. Although these protocols support multiple sinks, they do not identify path sharing nor do they optimize routes for multiple sinks. While the option of using a multicast tree has been discussed in [9], it assumes the network is IP-based and the multicast tree is built with off-line tools. Our most significant deviation from these works is our explicit exploration and learning mechanisms, allowing us to learn the parameters of the network with limited local exchange of information.

One WSN protocol for routing from multiple sources to multiple sinks recently appeared [10], however it merges initial local paths rather than searching for best paths in the network. It also assumes a perfect aggregation of packets from different sources. Subject-based networking in the mobile ad hoc domain is also related, e.g., MAODV [8]. While such an approach can be applied in sensor networks, the network properties and protocol requirements are significantly different.

The cornerstone of our protocol is its ability to learn better paths over time. It was partially inspired by AntHocNet [4], an approach for learning routes in wireless ad hoc networks based on ant colony optimization. In our case, however, the overhead of sending ants through the whole network would be considered a waste of energy. Second, and more significant, finding shorter routes to multiple sinks would imply the cloning of the ant to follow multiple paths, an option that would break the analogy and change the properties of AntHocNet.

A different form of learning, namely Reinforcement Learning, has been applied to different routing problems in networks [1, 2]. These works, however, ei-

ther assume global topology knowledge, or do not provide sufficient information regarding the communication protocol or how data should be exchanged.

5 Conclusion and Future Work

The concept of learning through feedback piggybacked on data packets is a powerful tool in WSNs as it requires limited local knowledge and achieves significant results. In this paper we presented the new concept of using feedback information to incrementally learn the routing properties of the network. The result is a new, efficient routing protocol that learns the best routes to multiple sinks.

Our future plans include further experimentation with new fitness functions and exploration strategies, and examining opportunities to exploit learning in different layers of the protocol stack as well as new domains including our work on non-uniform information dissemination.

References

1. 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, Paris, France*, 2005.
2. J. Boyan and M. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Advances in Neural Information Processing Systems*, volume 6, 1994, pp.671–678.
3. A. Carzaniga and A. L. Wolf, "Content-based networking: A new communication infrastructure," in *Proceedings of the NSF Workshop on an Infrastructure for Mobile and Wireless Systems, Scottsdale, AZ, USA*, 2001, pp. 59–68.
4. G. Di Caro, F. Ducatelle, and L. Gambardella, "AntHocNet: An ant-based hybrid routing algorithm for mobile adhoc networks," in *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature, Birmingham, UK*, 2004, pp. 461–470.
5. A. Egorova-Foerster, and A. Murphy, "A Feedback-Enhanced Learning Approach for Routing in WSN", Faculty of Informatics, University of Lugano, TR 2006/03, May 2006.
6. C. P. Hall, A. Carzaniga, J. Rose, and A. L. Wolf, "A content-based networking protocol for sensor networks," Department of Computer Science, University of Colorado, Tech. Rep. CU-CS-979-04, August 2004.
7. C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," in *Transactions on Networking*, volume 11, 2003, pp.2–16.
8. C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Proceedings of the 2nd IEEE Workshop on Mobile Computer Systems and Applications, Washington, DC, USA*, 1999.
9. F. Silva, J. Heidemann, R. Govindan, and D. Estrin, "Directed diffusion," Chapter in *Frontiers in Distributed Sensor Networks*, 2003, pp. 573–596.
10. P. Ciciriello, L. Mottola, and G. Picco, "Efficient routing from multiple sources to multiple sinks in wireless sensor networks," in *Proceedings of the 4th European Conference on Wireless Sensor Networks (EWSN)*, 2007, to appear.