Efficient Routing to Mobile Sinks in Wireless Sensor Networks

Kristóf Fodor, Attila Vidács
Dept. of Telecommunications and Media Informatics
Budapest University of Technology and Economics
H-1117 Budapest, Magyar tudósok krt. 2., Hungary
{fodork, vidacs}@tmit.bme.hu

ABSTRACT

In a typical wireless sensor network covering a large geographical area the range of the sensors' radio is in general quite short when compared to the network size. Thus, multihop communication is essential where nodes relay information packets between the source nodes and the sink(s). Because of the low-cost tiny devices the operation of the network is highly energy sensitive. The lifetime of the network largely depends on the energy of the sensor nodes neighboring the sink(s) that relay all messages on the last hop. Our solution to overcome this limitation is to use mobile sink(s) to move away from depleted areas. However, mobility raises several routing issues. In this paper we present a simple yet effective routing protocol, which uses restricted flooding to update the paths towards multiple mobile sinks in the network. The proposed solution tries to find a compromise between the optimal routes and the number of messages needed to update these routes. Routes are only updated where the degree of topology change requires it. We neither require sinks to subscribe for specific data in advance, nor need sink agents, and sinks are allowed to move on arbitrary paths.

Keywords

sensor network, routing, mobile sink

1. INTRODUCTION

Traditional wireless sensor networks (WSNs) consist of many small devices with limited energy resources and one or more sinks to which the sensors send their measurements. Sensor nodes are usually fixed and the permanent change in physical topology mostly comes from the disappearance of a node caused by the depletion of its battery.

Among others, sensor nodes differ from nodes in ad hoc network in the communication pattern: while ad hoc network nodes may contact any other node in the network, sensors nodes send data only to the sinks. The range of the sensors' radio is in general quite short, thus multihop communication is needed between source sensor nodes and the sinks. As a result, the volume of traffic in the network is focused in the neighborhood of the sinks. It follows that – in case of energy-homogeneous nodes in the network – nodes in the neighborhood of the static sinks are the first whose energy resource gets depleted. This also means that the lifetime of the network depends on these critical nodes in the vicinity of the sink. One solution to overcome this limitation is to use mobile sinks as proposed by some researcher in the last few years [1–10].

Mobile sinks may move on a fixed path [1,5,6] (e.g., sinks may be mounted on local buses running on schedule), may take a random walk (e.g., human's Personal Digital Assistant (PDA) serving as a sink [7]) or may move as instructed to optimal places in terms of network lifetime [10] (e.g., sinks carried by a mobile robot). Depending on the tolerated delay, sensor nodes may route the messages to the nearest sensor on the path of the sink (if the sinks moves along a fixed path) or may route the data directly to the sink, sometimes to the other end of the sensor network.

Several algorithms exist to determine the optimal placement and/or path of a sink. Furthermore, some routing mechanisms exist, which route data to mobile sinks moving along a fixed path, or moving along an arbitrary path but with the limitation that sinks subscrib in advance for the specific data or events they would like to get. However, to the best of our knowledge, there is no routing solution for WSNs that supports multiple mobile sinks with no restriction on their movement pattern and that does not require subscription for data in advance. In this paper we present a simple yet effective routing protocol, which uses restricted flooding to update the location of the mobile sinks in the network. The solution optimizes the traffic overhead arising from the route updates, and also minimizes the number of message types used for the communication.

We consider the network as a two-tiered architecture (Figure 1): on the first level mobile sinks communicate with each other and with the gateway of the network, while on the second level sensor nodes communicate with each other and with sinks. We do not differentiate between the mobile sinks: their role is only to forward measurements of sensor nodes to the gateway of the network, irrespective of their identity. Neither energy consumption of mobile sinks is considered in this paper, nor their optimal movement.

The proposed routing protocol tries to find a compromise between the optimal routes to the mobile sinks and the number of messages needed to update these routes. Only those routes or part of routes that forward data in a totally wrong direction and those that are far longer than the optimal mes-

WICON 2007, October 22-24, Austin, USA

DOI 10.4108/pwsn.2007.2266

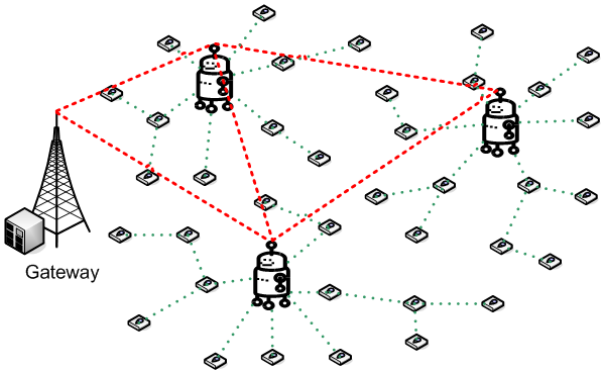


Figure 1: Two-tiered wireless sensor network architecture

sage path are updated.

The remainder of the paper is organized as follows. In the next section we provide a brief overview of the already existing routing protocols that support mobility of sinks. Then, we describe our routing algorithm: first, an algorithm that can be used in scenarios with only one mobile sink; then, an extension of this basic protocol that supports multiple mobile sinks in the network. Finally, we show simulation evaluation of a MATLAB implementation of our protocol. The last section concludes the paper.

2. RELATED WORK

In this section we provide a brief overview of routing algorithm, which support mobile sinks.

The first routing algorithms for sensor networks, like [13], did not consider the mobility of sinks at all. [13] builds up gradients pointing to the sinks by diffusing interests through the network. However there is no possibility to change these gradients significantly after setup. To integrate mobile sinks, the only solution to update the routes is to tear down the gradients before the mobile sinks moves away and set up new ones after the mobile sink stops.

[18, 19] propose to distribute traffic load in sensor networks by introducing virtual sinks equipped with an additional radio, which can be used to form a second tier on the top of the basic sensor network. Virtual sinks partition the network among themselves, and each of them becomes responsible for a specific region (ie., virtual sinks form clusters and act as clusterheads). Whenever virtual sinks receive a data sent from their region, they forward it to the real sink using the second tier network. Our solution is similar to these in terms of using heterogeneous nodes in the network. However we also consider sink mobility.

The first routing protocol that supports mobile sinks is TTDD [15], where each data source builds up an own grid structure. To get data from a source, sinks have to access one of the corresponding grid points through an Immediate Agent (IA) followed by a Primary Agent (PA) located in the actual grid cell. If the sink moves away, a new IA and PA have to be chosen depending on how far the sink moves.

[7] presents a similar concept to TTDD in terms of having an agent, called an access node, for data forwarding. This access node represents the mobile sink during operation. After having subscribed at a source, data updates are disseminated along a tree to the access node, which forwards it to the mobile sink. The tree is updated only if one of the access nodes changes, that is if one of the mobile sinks changes its access node. The switch between two access nodes may be triggered by the significant increase of hop distance between a sink and its access node. Our solution differs from these former two approaches, since we neither require sinks to subscribe for specific data in advance, nor need agents.

Some other routing protocols described in the following do not demand subscription for data in advance, however restrict the mobile sinks to move on a fixed path. When a sensor is triggered by an event, the node automatically sends the corresponding information to one of the mobile sinks in the network. [2, 3] propose a mechanism to send sensor data to the nearest node along the path of the sink. In the initialization phase all nodes discover the optimal route to the nearest point of the path. Then, in the second phase, nodes forward messages to the corresponding node located along the sink's path. These nodes pass the data to the mobile sink, called as data mule, when the sink approaches to them. The proposed solution is similar to the Delay Tolerant Network (DTN) [16] routing approach, since both mechanisms are based on the store-and-forward principle.

The authors of [1] suggest a reinforcement learning algorithm for sensor nodes that they call Hybrid Learning-Enforced Time Domain Routing (HLETDR). Each node continuously learns the movement pattern of the mobile sink and statistically characterize it as a probability distribution function. Thus, sensor nodes always know in which direction they have to route messages to the sink at a given time instant. The advantage of the solution is that nodes do not need time synchronization, since they make forwarding decisions in their local time-domain.

[5] takes scenarios where sensors are deployed within a circle. The authors argue that in such cases the mobile sink should follow the periphery of the network in order to optimize the energy consumption of the nodes. They also show that an efficient routing algorithm routes messages directly to the sink – on a shortest path – if the sender is closer to the center of the circle than the sink. However, nodes deployed in the outer annulus related to the sink route messages first around the center of the circle till they cross the line defined by the circle center and the sink (clockwise or counterclockwise, depending on how the forwarding node and the sink are positioned relative to each other). Then, the message is forwarded on the shortest path towards the sink. The path or position of the sink may be preprogrammed in the sensor nodes or may be advertised by flooding the network whenever the sink moves away.

In contrast to the algorithms proposed for routing data to a mobile sink following a fixed path, in our solution we allow sinks to move on arbitrary paths, necessitating the design of new routing protocols.

Finally, the MobiRoute solution [6] extends the Berkeley MintRoute [17] by adding mobile sink support to the distance-vector based routing protocol. The sink is required to move between several anchor points at which the sink stops for a longer time (than the time elapsed during movement). The movement of the sink between two anchor points is broken down into four phases: pause, pre-move, move and pre-pause stages. MintRoute is used for routing in the network. However, in the pre-pause state, after the mobile sink stops, the routes have to be quickly updated. These updates

are done by MobiRoute by sending route update messages at a high rate. In our proposal, we do not limit mobile sinks by requiring long pause states in order to decrease topology updates. Instead, we allow their continuous movement, but we update only those routes or parts of routes where the degree of topology change requires it.

3. ROUTING TO MOBILE SINKS

3.1 One Sink in the Network

In the first scenario, we assume a network consisting of hundreds or thousands of sensor nodes and one mobile sink, which moves continuously on an arbitrary unknown path. The position of the sink cannot be determined in advance, it depends on a black box algorithm considering this paper (e.g., the algorithm presented in [12] could be used). Sensor nodes have limited radio range, thus multihop communication is used in the network. Nodes neither have globally unique network identifiers, nor can they be addressed based on their geographic location. Events to be reported to the sink may occur at each node with the same probability; there are no differentiated groups of sensors or physical locations of event occurrences. Furthermore, we do not consider data aggregation in the network: each event is be published individually to the sink.

We consider using a gradient-based routing approach in the network, where each node maintains a list of neighboring sensor nodes that are in right direction towards the sink. When nodes get a message to forward to the sink or are triggered by an event, they pass the message to one of their neighbors in this list. An easy solution to get continuously valid routes towards the actual position of the sink is to have the mobile sink to periodically flood the network. Then, based on the flooding, a tree can be built up with the mobile sink as the root. The algorithm supporting this approach would be quite simple: the mobile sink locally broadcasts its presence periodically and each sensor that gets an update message with a higher sequence number than registered in the past registers the sender as its new parent. Then, these nodes forward the message to their neighbors. Nodes receiving a message with an already registered sequence number discard the message. Although this solution could be an answer for the problem to maintain valid routes to the sink, it would induce a very heavy traffic load in the network. In consequence, this would result a very short network lifetime, since sensor nodes have only limited energy resources.

In order to extend the network lifetime, we propose to modify the above presented basic routing protocol and update only those parts of the tree that are far from optimal in terms of forwarding cost and load distribution. Since nodes in the vicinity of the sink are the most loaded by the traffic, the effect of the sink movement should be most perceived by these nodes. Furthermore, nodes farther away from the sink may be updated less regularly than nodes closer to the sink. Since these far-away nodes send their message through many hops, a slightly higher total forwarding cost along the route – because of an eventual non-up-to-date topology knowledge – does not impose a significant performance degradation compared to the gain we get from using fewer routing updates.

The basic principle of our routing protocol is to register a cost between the sink and each sensor node, and update only those routing entries where the change in the cost is above a threshold. The cost function may be chosen arbitrarily, e.g. the cost may be the distance (number of hops) from the sink, which we have used, or some quality of service related metric. The costs are propagated in the update messages. We define the thresholds related to the ratio of the registered cost and the actual cost propagated by the neighbors. If the difference of costs exceeds a given percentage, the cost is updated, otherwise the update message is discarded. The cost update not only changes the costs at the nodes forming the tree, but may also result changes in the structure of the tree.

One of the main consequences of the sink movement is that the routing tree continuously gets partitioned. Since we use a tree topology for routing with the sink as its root, whenever a sink moves away from a neighboring sensor node, the branch connected to that neighbor gets disconnected. Figure 2 shows such a situation: after the sink has changed its position (time instant *t2*) nodes that route their messages through node *A* (highlighted in the right picture) cannot contact the sink any more until the disconnected branch is reconnected to the rest of the network. Thus, we not only have to optimally update the valid routes based on a cost function, but we also have to handle partitionings caused by the movement of the sink.

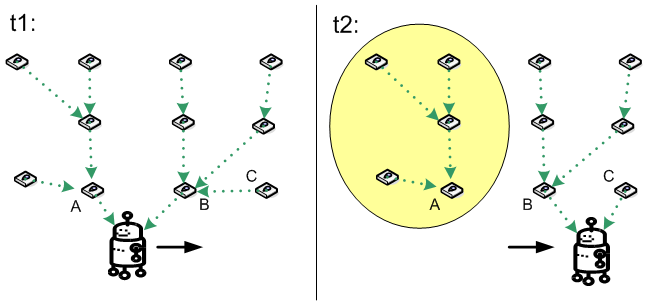


Figure 2: Left branch of the routing tree splits off after the mobile sink moved away from node *A*

Branches split off from the routing tree due to the movement of the sink should be reconnected as soon as possible. We use an expanding ring search method for this purpose. Immediately after the ex-neighbor node of the sink detects the disappearance of the mobile sink, it broadcasts a discovery message to its neighbors (i.e., does a 1-hop flood) including the identifier of the mobile sink and the sequence number of the last received update message. Moreover, it sets its cost parameter to infinite. Neighboring nodes answer for this request by broadcasting an update message if they have a routing entry corresponding to the mobile sink with a higher sequence number. If no one returns a routing direction to the sink (i.e., the initiator gets no update message with a higher sequence number) in a certain timeframe, then the ex-neighbor of the sink restarts the flooding but targeting nodes in a 2-hop distance this time. If the requester still does not receive any answer, then it increases the number of hops until its discovery messages either reach a node with more up-to-date information on the base station's location or reach the base station, or the number of maximal hops exceeds a certain limit.

Nodes always set their cost to infinity when they receive a discovery message and have no newer information on the lo-

cation of the mobile sink than the requester. However, they retain the entry on the next hop towards the last known position of the mobile base station. This way disconnected tree branches can be reconnected easily by reusing the standard update messages – no new message type has to be introduced. In this way, all nodes between the root of the disconnected tree branch and the node that has newer information will have an infinite cost to the mobile sink assigned. Since routing entries with infinite cost are always updated if receiving an update message containing a finite cost, these entries will be certainly updated if at least one node in the network has newer information. The route between the root of the disconnected tree branch will be set up by updating the intermediate nodes one after the other starting from the neighbor of the node that replies to the discovery message. Furthermore, by not removing next hop entries at nodes with infinite cost we can avoid needlessly dropped packets after the ex-neighbor of the sink is reconfigured, but yet not all the nodes whose cost was set to infinity during the expanding ring search.

In order to decrease the outage times when a branch is temporally disconnected, the ratio of the speed of the mobile sink and the frequency of the beacons emitted by the mobile sink – to signal its presence – has to be selected carefully. However, even if we define the highest speed very carefully, in some scenarios the movement of the sink may be such that some nodes cannot reach the mobile sink for a longer time. Let us consider an example as depicted in Figure 3. At time instant *t1* the sink is connected to node *A* and all the other nodes in the same branch forward data through this node. The cost of sending data is equal to the number of hops between the given node and the sink (in the figure, the number next to the nodes shows the cost). Even if we immediately update a path when there is a slightly better route with lower cost (that is, we use the lowest possible threshold when updating the routes) at time instant *t2*, after the sink has moved next to node *G*, only half of the *G-F-E-D-C-B-A-sink* path can be updated. This is because the cost of data forwarding registered at the nodes increases with the distance from the sink, and after a while the update messages reach a node that registers a lower cost. Although this lower cost is not valid any more (since the path does not ends by the sink), the given node is not aware about this. The path between node *A* and node *D* in the figure can only be torn down and set up again (but from the other direction) after node *A* has realized that the mobile sink moved away, i.e., a timer assigned to the beacons of the mobile sink has expired. Of course, both this realization phase and then the expanding ring search take some time, which means that the concerned nodes in the branch cannot reach the sink temporarily.

3.2 Multiple Sinks Present in the Network

In the second scenario, we envisage the same situation as in the first scenario, but with more than one sink (Figure 1). We do not differentiate the individual mobile sinks; sensor nodes can forward data to any of them, since all the sinks forward the data further on to the gateway of the network, which then process it.

Our basic routing protocol that supports multiple mobile sinks requires sensor nodes to register only one route to one of the base stations. When updating routes the identity of the base stations is not considered; the forwarding direction

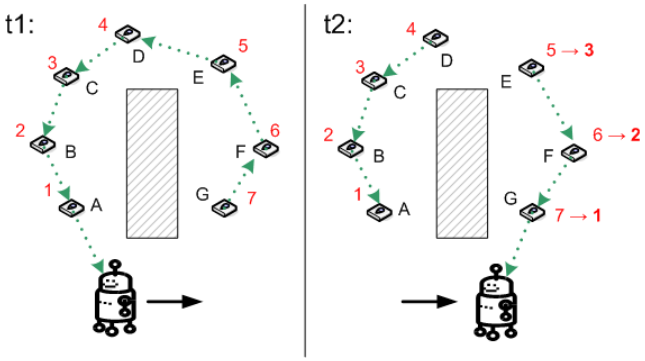


Figure 3: Unlucky scenario – nodes *A*, *B*, *C* and *D* will get temporarily disconnected at time instant *t2*

is selected purely based on the costs included in the update messages. The benefit of having multiple mobile sinks in the network is the decrease of average route length.

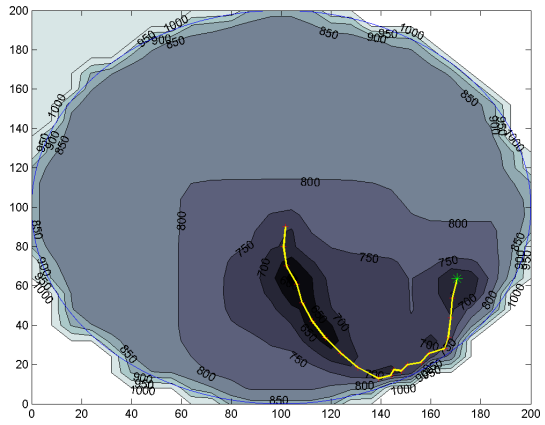
To improve the above protocol, we can register more than one route at a sensor node, each pointing to a different mobile sink. Basically, the first entry (the entry corresponding to the lowest registered cost) is used for routing – the others are kept as backups. Whenever a branch of the tree gets disconnected (because the sink has changed its position), the sensor nodes can switch to another path and forward data to another mobile sink. Using this protocol, the period when nodes are temporally disconnected can be significantly minimized. The details of the updating mechanism are presented in Appendix A and B.

When registering multiple mobile sinks at the nodes, the number of the maximal routing entries per node has to be chosen carefully. One may think that registering more backup paths leads to lower average route disconnection durations. However, we also have to take into account that the routes require continuous updates. This increases the traffic load, which in turn decreases the network lifetime. Thus, a node should maintain only as many routing entries as are necessary to guarantee that the average disconnection period of a node is below the given level.

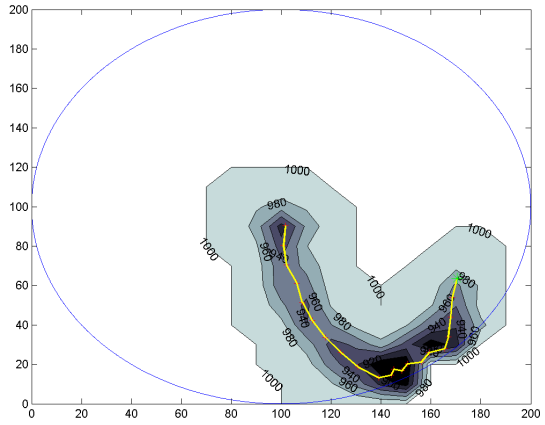
4. EVALUATION

In order to prove the efficiency of our algorithms, we have run simulations in MATLAB [14]. In the simulations we look into the first scenario (described in Section 3.1), which contains 1000 nodes placed uniform randomly and one mobile sink. Nodes have 6 neighbors on average. Energy needed both for sending and receiving a message is considered as 1 unit, while the costs used at route updates reflect the number of hops between the sink and the actual sensor node. During the analysis the mobile sink moves randomly with a constant speed in a circle shaped field. However, we limit the direction deviation of two consecutive movements. In the simulations, we investigate the number of messages our routing algorithm sends and receives, and the energy consumption of the nodes. The results are compared to the reference model, where the network is flooded with routing update messages whenever the mobile sink changes its position.

Figure 4 shows two heat maps: we depict the traces of the



(a) Routes were updated in case of minor change in distance between node and mobile sink



(b) Routes were updated only in case of major change in distance between node and mobile sink

Figure 4: Remaining energy of sensor nodes after moving a sink the sink on a short path of 20 steps

mobile sink, and the remaining energy levels of the nodes in the networks after the sink moves along a short path. (The default energy level of the nodes is set to 1000.) The two scenarios differ in the threshold limit, which specifies when a route has to be updated (when the update message has to be forwarded). As it can be seen in both figures, nodes farther away from the path of the sink have more remaining energy than nodes closer to it. This is because we keep the routing updates as local as possible and avoid forwarding the update messages to areas where it is not necessary.

We may also see that nodes in Figure 4(a) consumed more energy than nodes depicted in Figure 4(b). This comes from the difference in the update threshold: while in the later scenario we updated the routes only if they were far from optimal, in the former scenario we also updated routes close to the best possible length.

Since the frequency of route updates not only affects the energy consumption of nodes, but is also in strong relationship with the route lengths, we investigate the connection between average route lengths and routing update thresholds. We seek reducing the energy consumption needed for route updates if the energy used to forward data messages (reports of the sensors) increases faster than the energy used for topology control decreases.

We run several series of simulations with different thresholds used for routing updates (see Figure 5). The lowest threshold value is 0%, which means that the routes are always updated if there is a minor change in the cost registered at the actual node towards the mobile sink. This scenario is similar to the reference model, in which routes are always updated when the mobile sink moves away. The only difference is that ex-neighboring nodes of the mobile sink have to start an expanding ring search to reconnect the disconnected branch to the rest of the network if they notice the departure of the mobile sink earlier than they receive an update message. The consequence of this difference can be seen in Figure 5 at 0%: although the average route lengths are equal in the reference model and when using our algorithm, our mechanism uses more energy to reconfigure the routes. This means our solution should not be used when data is always required to be sent along the optimal route (in terms of length) towards the mobile sink.

However, Figure 5 also shows that energy consumption can be significantly reduced by setting even a small non-zero threshold value. Depending on the threshold value, the energy gain can be 78 - 95%. Since the average length of routes also increases with the energy gain, the threshold should be set carefully.

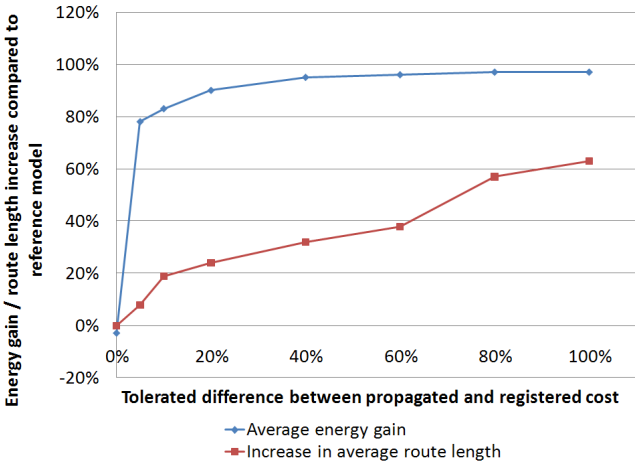


Figure 5: Average route length vs. average energy gain compared to reference model

We propose to let the mobile sinks set the routing update threshold dynamically: the update messages shall include an extra field for the actual threshold value. When the sink detects that nodes send data more frequently (e.g., an alarm is triggered) it should lower the threshold propagated in the update messages, and when it gets less packets from the nodes (e.g., in idle state of the network, when nodes do not send data for a longer period), it should raise it. This way, energy consumption of nodes due to forwarding data and updating routing entries can be optimized together.

5. CONCLUSION

In this paper we have proposed a new routing solution in wireless sensor networks that supports multiple mobile sinks. We assume a multi-hop network with a high number of nodes, where the data gathered must be sent to dedicated sink nodes. Nodes do not have network-wide unique identifiers, and they can not be addressed based on their geographic location. We do not differentiate between the sinks; all nodes can choose freely to which one of them they send the data. Our routing solution is a gradient-based approach, where nodes maintain a list of neighboring 'next-hops' that are in right direction towards the closest sink. The protocol uses restricted flooding to update the location of the mobile sinks while trying to compromise between optimal routing and the signalling overhead. The basic principle behind is to register a cost between the appropriate sink and the given node at each node, and update only those routing entries where the relative change in the cost is above a threshold. Our basic solution requires sensor nodes to register only one route towards only one of the sinks. The sink identifier is not considered and neither propagated, the forwarding decision is made purely based on the cost included in the update message. We have presented an extension to this protocol as well to increase robustness. Here the sensor nodes register routes and weights to more than one mobile sinks.

Finally, we have evaluated our protocol for the scenario with one mobile sink. Simulation results obtained with MATLAB show that up to 95% energy gain can be achieved with our solution compared to the basic protocol used as reference, which always floods the network when the position of the mobile sink changes. Although the average route length also increases when doing less update, the total energy consumption needed for forwarding data and reconfiguring routes may be reduced by changing the update value dynamically. In the future we plan to analyze the performance of our solution in the case of multiple sinks present in the network.

6. REFERENCES

[1] P. Baruah, and R. Urgaonkar, B. Krishnamachari: *Learning-Enforced Time Domain Routing to Mobile Sinks in Wireless Sensor Fields*, in Proc. of 29th Annual IEEE International Conference on Local Computer, Tampa, FL, USA, November 2004.

[2] A.A. Somasundara, A. Kansal, D.D. Jea, Jea, D. Estrin, and M. B. Srivastava: *Controllably Mobile Infrastructure for Low Energy Embedded Networks*, in IEEE Transactions on Mobile Computing, Vol.5, No.8, pp. 958-973, August 2006.

[3] D. Jea, A. A. Somasundara, M. B. Srivastava: *Multiple Controlled Mobile Elements (Data Mules) for Data Collection in Sensor Networks*, in Proc. of 1st IEEE/ACM International Conference on Distributed Computing in Sensor Systems (DCOSS '05), June 2005.

[4] Shashidhar Gandham, Milind Dawande, Ravi Prakash, Subbarayan Venkatesan: *Energy Efficient Schemes for Wireless Sensor Networks with Multiple Mobile Base Stations*, in Proc. of the IEEE Global Telecommunications Conference 2003 (Globecom 2003), San Francisco, CA, USA, December 2003.

[5] J. Luo and J.-P. Hubaux: *Joint Mobility and Routing for Lifetime Elongation in Wireless Sensor Networks*, in Proc. of the 24th The Conference on Computer Communications (Infocom 2005), Miami, FL, USA, March 2005.

[6] J. Luo, J. Panchard, M. Piorkowski, M. Grossglauser and J.-P. Hubaux: *MobiRoute: Routing towards a Mobile Sink for Improving Lifetime in Sensor Networks*, in Proc. of 2nd IEEE/ACM International Conference on Distributed Computing in Sensor Systems (DCOSS'06), San Francisco, CA, USA, June 2006.

[7] Hyung Seok Kim, Tarek F. Abdelzaher, Wook Hyun Kwon: *Minimum-Energy Asynchronous Dissemination to Mobile Sinks in Wireless Sensor Networks*, in Proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys 2003), Los Angeles, CA, USA, November 2003.

[8] Z.M.Wang, S. Basagni, E. Melachrinoudis, C. Petrioli: *Exploiting Sink Mobility for Maximizing Sensor Networks Lifetime*, in Proc. of the 38th Hawaii International Conference on System Sciences (HICSS-38 2005), Big Island, Hawaii, USA, January 2005.

[9] Z. Vincze, D. Vass, R. Vida and A. Vidács, A. Telcs: *Sink Mobility in Event-driven Multi-hop Wireless Sensor Networks*, in Proc. of the 1st International Conference on Integrated Internet Ad hoc and Sensor Networks (InterSense), Nice, France, May 2006.

[10] Z. Vincze, K. Fodor, R. Vida, A. Vidács: *Electrostatic Modelling of Multiple Mobile Sinks in Wireless Sensor Networks*, in Proc. of the IFIP Networking Workshop on Performance Control in Wireless Sensor Networks (PWSN 2006), Coimbra, Portugal, May 2006.

[11] S. Jain, R. Shah, W. Brunette, G. Borriello, S. Roy: *Exploiting Mobility for Energy Efficient Data Collection in Sensor Networks*, Mobile Networks and Applications, Springer Science, Vol. 11, No. 3, June 2006.

[12] Z. Vincze, D. Vass, R. Vida, A. Vidács, A. Telcs: *Adaptive Sink Mobility in Event-Driven Densely Deployed Wireless Sensor Networks*, International Journal on Ad Hoc & Sensor Networks, 2007.

[13] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva: *Directed Diffusion for Wireless Sensor Networking*, in IEEE/ACM Transactions on Networking, Vol. 11, No. 1, pp. 2-16, February 2003.

[14] Matlab simulator, http://www.mathworks.com/products/matlab/

[15] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang: *A Two-tier Data Dissemination Model for Large Scale Wireless Sensor Networks*, in Proc. of the 8th ACM Annual International Conference on Mobile Computing and Networking (MobiCom 2002), Atlanta, GA, USA, September 2002.

[16] K. Fall: *A Delay-Tolerant Network Architecture for Challenged Internets*, in the Proc. of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, Karlsruhe, Germany, August 2003.

[17] A. Woo, T. Tong, and D. Culler: *Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks*, in the Proc. of the 1st ACM SenSys, Los Angeles, CA, USA, November 2003.

[18] Chieh-Yih Wan, Shane B. Eisenman, Andrew T. Campbell, and Jon Crowcroft: *Siphon: Overload Traffic Management using Multi-Radio Virtual Sinks*, in Proc. of Third ACM Conference on Embedded Networked Sensor Systems (SenSys 2005), pp. 116-129, San Diego, November 2005. Springer International Journal of Wireless Information Networks, Vol. 12, No. 1, Jan 2005.

[19] W. Hu, N. Bulusu, and S. Jha: *Overload Traffic Management for Sensor Networks*, ACM Transactions on Sensor Networks, Vol. 3, No. 4, October 2007.

APPENDIX

A. UPDATING ROUTING ENTRIES

Algorithm A.1: ReceiveRoutingUpdateMessage($cost\_propagated, ms\_id, seq\_num$)

```
procedure UpdateRoutingEntries(ms_id, cost)
 if IsMSRegistered(ms_id)
   then { ModifyRoutingEntry(ms_id, cost)
          BroadcastRoutingMessage(cost+
                                  forwading_cost, ms_id)
   else if GetNumOfRoutingEntries() <
     max_entries_num
   then { AddRoutingEntry(ms_id, cost)
          BroadcastRoutingMessage(cost+
                                  forwading_cost, ms_id)
   else if GetHighestRegisteredCost() > cost
   then { RemoveRoutingEntry(
              GetMSWithHighestCost())
          AddRoutingEntry(ms_id, cost)
          BroadcastRoutingMessage(cost+
                                  forwading_cost, ms_id)

main
 if cost_propagated = 0
   then UpdateRoutingEntries(ms_id, 0)
   else if IsBSRegistered(ms_id) and
           GetRegCostToMS(ms_id)! = infinite
   then { if |GetRegCostToMS(ms_id)−
                                  cost_propagated)| >
          GetRegCostToMS(ms_id) ∗ threshold_percent
           and
          GetLastSequenceNumber(ms_id) < seq_num
            then UpdateRoutingEntries(ms_id,
                                      cost_propagated)
   else UpdateRoutingEntries(ms_id, cost_propagated)
 UpdateSequenceNumbersRegistry(ms_id, seq_num)
```

B. RECONNECTING DISCONNECTED BRANCH OF THE ROUTING TREE

Algorithm B.1: ReconnectMSExNeighbor()

```
procedure MSNotInRadioRangeAnymore(ms_id)
 max_hop_num = 1;
 BroadcastMSSearchMessage(ms_id,
   GetLastSequenceNumber(ms_id), max_hop_num − 1)
 ResetMSSearchTimer()

procedure ReceiveMSSearchMessage(ms_id,
                                 last_seq_num, hops_left)
 if IsMSRegistered(ms_id) and
 GetLastSequenceNumber(ms_id) > last_seq_num
   then BroadcastRoutingMessage(
     GetRegCostToMS(ms_id) + forwading_cost, ms_id)
   else { ModifyRoutingEntry(ms_id, infinite)
          if hops_left > 0
            then BroadcastMSSearchMessage(ms_id,
                                 last_seq_num, hops_left − 1)

procedure MSSearchTimerExpires(ms_id)
 if GetRegCostToMS(ms_id) = infinite
   then { IncreaseMaxHopNum(max_hop_num)
          BroadcastMSSearchMessage(ms_id,
                 GetLastSequenceNumber(ms_id),
                                 max_hop_num − 1)
          ResetMSSearchTimer()
```