Routing towards a mobile sink using virtual coordinates in a wireless sensor network

Rouhollah Rahmatizadeh*, Saad Ahmad Khan*, Anura P. Jayasumana†, Damla Turgut* and Ladislau Bölöni*

*Department of Electrical Engineering and Computer Science
University of Central Florida, Orlando, FL
Email: rrahmati@knights.ucf.edu, {skhan, turgut, lboloni}@eecs.ucf.edu
† Department of Electrical and Computer Engineering,
Colorado State University, Fort Collins, CO
Email: anura.jayasumana@colostate.edu

***Abstract*—Geographical routing can provide significant advantages in wireless sensor networks. However in many sensor networks, it is difficult or costly to find the exact location of the nodes. The virtual coordinate techniques allow a network to acquire a coordinate system without relying on geographical location. In this paper, we describe MS-DVCR, an extension of a state-of-the-art virtual coordinate routing protocol (DVCR) with the ability to route towards a mobile sink. We describe the design principles and implementation of the proposed protocol and through an experimental study, we show that it matches the performance of a simple extension of DVCR for mobile sinks while providing a significantly lower energy consumption.**

I. Introduction

The use of a coordinate system in a wireless sensor network allows a source node to infer the approximate direction of the destination without the need of a full picture of the network. The most often used coordinate system relies on the actual geographical coordinates of a node [1]. This information, however, is difficult to acquire in many practical sensor networks. For instance, cost considerations prevent equipping every node with a GPS unit, or the network may be in a location such as indoors where the GPS signal is not available. As an alternative, it is possible for a network to build a *virtual coordinate* (VC) system based on hop-by-hop distances from a few anchor points [2], [3], [4].

Recent research has shown that VCs can be used to create topology preserving maps [5] which can be used instead of geographic coordinate based maps of networks. Directed VCs [6] add directionality to the concept of virtual coordinates. DVCs provide most of the advantages of geographical routing, without the requirement of having the exact location of each node.

Early work in sensor networks assumed the existence of a single sink node with a fixed position. However, many sensor networks need to consider mobile sinks, either due to external requirements [7] or as a specific design choice. The movement of the mobile sinks may range from random [8], [9], [10] to regular and predictable [11].

As of yet, there is no work in the literature which provides routing towards a mobile sink in the context of DVCs. Naturally, it is always possible for a mobile sink to broadcast its new coordinates to the entire network whenever it moves. This, however, is a solution with high overhead in terms of messages and energy consumption. The objective of this paper is to design a routing protocol which allows the efficient routing towards a mobile sink in a directional virtual coordinate system. The approach we had taken is to integrate into the DVCR routing protocol [6] some of the ideas which had been successfully applied to geographical routing protocols [12] to yield the Mobile Sink DVCR (MS-DVCR) protocol.

The remainder of this paper is organized as follows. Section II explains the geometrical construction and calculations underlining a virtual coordinate system, and discusses directional virtual coordinates and the DVCR routing algorithm. The MS-DVCR algorithm is presented in Section III. We validate the algorithm using a series of simulation studies in Section IV and conclude in Section V.

II. Virtual coordinates

A. General principles

Lets us consider a sensor network with N sensor nodes. We choose a small set of M nodes $A_1 \dots A_M$ as *anchor nodes*. We will define the virtual coordinates of a node as a vector of M numbers, where each number represents the number of hops between the node and the corresponding anchor. There are several ways in which the nodes can find their own coordinates in the network. One approach is to use a network setup phase using network-wide flooding [3]. Alternatively, the nodes can find the coordinates gradually without any need for a setup phase using Rumor Routing [13].

Just like geographical coordinates, VCs have been used to guide a routing algorithm by providing an estimation of the distance between the source and destination node. The distance metrics in VCs are based on the L^1 or L^2 norms defined over VCs. Just like in the case of geographical coordinates, networks of a non-convex shape can lead messages following the distance metric into a local minima. Thus, VC routing protocols usually deploy techniques to overcome the local minima problem using an estimation of distance to destination. As the VCs propagate radially away from the anchors, the L1 and

L2 do not provide good estimates of distance in VC systems, causing many local minima and poor routing performance. Therefore, new coordinates are derived from VCs to overcome this disadvantage and generate more Cartesian like coordinate systems. Directional Virtual Coordinate System (DVCS) is such an example.

B. Directional Virtual Coordinate System Routing

We have taken as the basis of our work, the directional virtual coordinate routing protocol (DVCR) [6]. Compared to earlier virtual coordinate routing protocols where the directionality of coordinates is lost, DVCR applies a mathematical transformation to restore the directionality of the ordinates.

Lets consider a one dimensional virtual coordinate system, where two anchors A_1 and A_2 are $h_{A_1A_2}$ hops apart. Mathematically, $h_{N_iA_1} - h_{N_iA_2}$ provides partial sense of directionality but has constant value outside of the region bounded by the anchors. For covering the exterior regions that are bounded by anchors, we need another constant, i.e. $h_{N_iA_1} + h_{N_iA_2}$ that varies linearly in the exterior regions bounded between both anchors. Therefore, the coordinate of a node with reference to anchor node A_1 and A_2 is given as

$$f(h_{N_iA_1}, h_{N_iA_2}) = \frac{1}{2h_{A_2A_1}}(h_{N_iA_1} - h_{N_iA_2})(h_{N_iA_1} + h_{N_iA_2}) \tag{1}$$

where $\frac{1}{2h_{A_2A_1}}$ is used for normalization. Let $\vec{f}(h_{N_iA_j}, h_{N_iA_k})$ be the vector representation using ordinates with reference to anchor A_j and anchor A_k. Therefore the vector is given as

$$\vec{f}(h_{N_iA_j}, h_{N_iA_k}) \quad = \quad f(h_{N_iA_j}, h_{N_iA_k})\vec{u}_{A_jA_k} \tag{2}$$

where $\vec{u}_{A_jA_k}$ is called the virtual direction and is the unit vector in direction of A_jA_k. The magnitude of the vector $f(h_{N_iA_j}, h_{N_iA_k})$ is given as

$$f(h_{N_iA_j}, h_{N_iA_k}) \quad = \quad \frac{1}{2h_{A_jA_k}}\left(h^2_{N_iA_j} - h^2_{N_iA_k}\right) \tag{3}$$

Let us now introduce a definition of the distance metric used in DVCR. Let suppose that the transformed ordinates of the source node x and the sink node y are given as $N_x \equiv [n_{x1} \cdots n_{xy} \cdots n_{xP}]$ and $N_y \equiv [n_{y1} \cdots n_{xy} \cdots n_{yP}]$. Here P is the cardinality of the ordinates and can be selected from C_2^M combinations given M randomly select anchors. Using the L^2 distance between the source and destination node we find the distance as

$$D_{N_xN_y} = \sqrt{\sum_{\forall j}(n_{xj} - n_{xy})^2}; j = 1 : J \leq C_2^M \tag{4}$$

This distance metric allows us to perform *greedy forwarding:* when a node needs to transmit a message to the destination (usually, the sink), it will forward the message to the neighbor which is closest to the destination in terms of the defined distance D.

Although greedy forwarding works fine in many situations, it is prone to the local minima problem in networks with a concave shape or networks with holes. To avoid messages getting stuck in a local minima, DVCR uses the ordinate difference between the nodes and its neighbors. Let us consider the ordinate difference set $\Delta_{A_1A_2}$ with reference to anchor nodes A_1 and A_2. Therefore,

$$\Delta_{A_1A_2} = |(F_{A_1A_2}(N_i, N_k)|; N_k \in K \tag{5}$$

where K is the total number of neighbors of node N_i. Let the maximum ordinate difference be α_{12} = max($\Delta_{A_1A_2}$) and the minimum ordinate difference be β_{12} = min($\Delta_{A_1A_2}$). Therefore, the approximate ordinate difference between current node and destination is given as:

$$\alpha_{12}n + \beta_{12}m = |F_{A_1A_2}(N_i, N_d)| \tag{6}$$

Similarly using reference anchor nodes A_3 and A_4 we get,

$$\alpha_{34}n + \beta_{34}m = |F_{A_3A_4}(N_i, N_d)| \tag{7}$$

By solving Equations 6 and 7, we are able to find $n + m$ which gives us an estimate of minimum number of hops to the destination. Similar calculations are performed by all of the neighbors of current node i and the node having the minimum number of hops is selected for forwarding.

III. The MS-DVCR routing protocol

Let us now consider the use of DVCR in a sensor network with static nodes and a single mobile sink. If the sink moves, its virtual coordinates will change, and the messages routed to the old coordinates will not reach the sink. A simple solution would be to notify all the nodes about the sink's new coordinates. This solution, however is expensive in terms of the number of messages, and the corresponding energy consumption.

The MS-DVCR algorithm takes an idea which had been successfully applied to geographical routing [12] to reduce the number of update messages necessary to maintain routability.

The general idea is that as long as the sink moves inside a limited *local area*, the nodes outside that area will not be notified about the sink's movement. The routing will rely on the nodes at the periphery of the area to guide the messages to the sink.

Let us start by defining the shape of the local area. Let us consider that at the time of the creation of the local area L the sink is at the location $N_s^c = [n_{s1}^c \ldots n_{sP}^c]$. Thus, the local area will be defined as all the nodes which are closer than L2 distance r to the initial location of the sink:

$$R = \left\{ N \;\middle|\; D_{N,S} = \sqrt{\sum_{i=1}^{P}(n_{si}^c - n_{Ni})^2} \leq r \right\} \tag{8}$$

Note, however, that the current location of the mobile sink might be different from N_s^c. Defining the local area, we say that the sink can make two different types of moves:

- a **local move** keeps the sink inside the local area. In this case, the sink will update only the nodes inside the local area about its new location, and the local area will not change.
- in an **external move** the sink leaves the current local area R. As a result, the sink must (a) create a new local area R' (see Figure 1) and (b) notify the whole network about its new virtual coordinates.

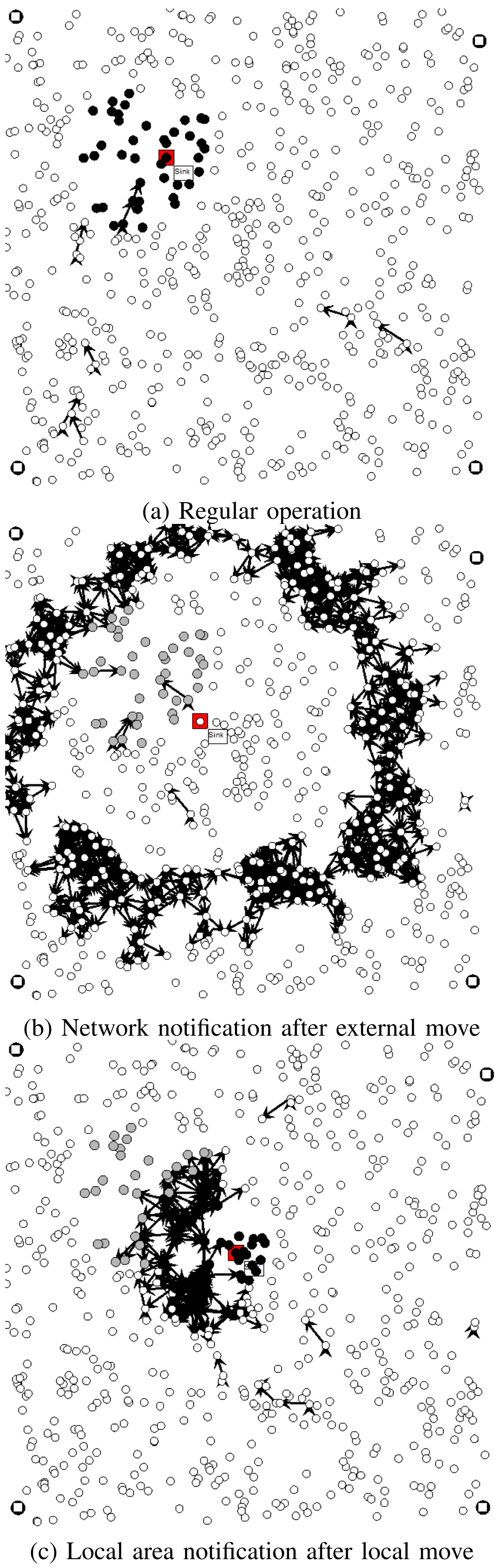
MS-DVCR uses three type of messages: (a) LOCAL messages carry updates about the local moves of the sink and they are broadcasted only within the confines of the local area R, (b) EXTERNAL messages carry updates about the external move of the sink and they are broadcasted to the whole area and (c) SENSING messages which carry data collected by the network, and are transmitted by hop-by-hop transmission from the nodes to the sink (whichever its current location it may be).

Algorithm 1 describes the event-driven behavior of the sink in MS-DVCR. The sink maintains the location of the center of the current local area. Whenever it moves, the sink will determine whether it left the current local area. If no, it will send a LOCAL message and keeps the local area center. If it left the local area, the sink will change the location of the center to its current location and send an EXTERNAL broadcast. In addition, the sink also receives SENSING messages which it collects and stores or processes on a domain-dependent basis.

Algorithm 1 Sink behavior in MS-DVCR

```
when move do
  new-location := current location of sink
  if ($D_{L2}$(new-location, local-area-center)) < r then
    broadcast(msg(LOCAL, new-location))
  else
    local-area-center := new-location
    broadcast(msg(EXTERNAL, local-area-center))
  end if
end when
when receives(msg(SENSING, data)) do
  update local model with data
end when
```

Algorithm 2 describes the event-driven behavior of the nodes in MS-DVCR. The nodes reach differently when receiving a LOCAL or an EXTERNAL message. When receiving a LOCAL message, the node needs to decide whether it is inside the new local area. It can do this by calculating the distance in directed virtual coordinates between the center of the new local area and itself. If this distance is smaller than the radius r, the node is in the area and it will re-broadcast the message. Otherwise, the node is on the border of the local area and it will not re-broadcast the message. Either way, the node uses the message to update the next hop it will use to forward the SENSING messages to. In the case of receiving an EXTERNAL message, the node will also update the next hop, which in this case it will be also the new



(a) Regular operation

(b) Network notification after external move

(c) Local area notification after local move

Fig. 1. (a) Operation of MS-DVCR between sink moves, (b) Network update after an external move (c) Network update after a local move. (black nodes: current local area, grey nodes: previous local area, thick circles in the corner: anchor nodes)

TABLE I
EXPERIMENTAL PARAMETERS

Parameter	Values
General	
Sensor network size $L \times L$	400…1000 m
Node deployment	random uniform
Deployment density	0.004 nodes / m^2
Number of sensor nodes	640-4000
Transmission range	30 m
Sink movement	random waypoint, no stops
Sink movement trigger	after 20 received messages
Experiment length	4000 messages
Protocols	
Routing protocol	UA-DVCR, MS-DVCR
Coordinates	directed virtual
Anchors	4, extreme corners
local area radius	$r \in \{2, \ldots, 12\}$

center of the local-area, and it will always re-broadcast the message. When receiving SENSING type messages, the nodes will forward them by sending them to the next hop set through the processing of the LOCAL or EXTERNAL messages.

Algorithm 2 Node behavior in MS-DVCR

```
when receives(message(LOCAL, new-sink-location)) do
    nexthop := closest neighbor to new-sink-location
    if ($D_{L2}$(local-area-center, nodelocation) $<$ r) then
        broadcast(msg(LOCAL, local-area-center))
    end if
end when
when receives(message(EXTERNAL, local-area-center)) do
    nexthop := closest neighbor to local-area-center
    broadcast(msg(EXTERNAL, local-area-center))
end when
when receives(message(SENSING, data)) do
    send(msg(SENSING, data), nexthop)
end when
when sensor-captures(observation) do
    data = report-formation(observation)
    send(msg(SENSING, data), nexthop)
end when
```

IV. Experimental study

A. Experimental setup

To study the properties of the MS-DVCR algorithm, we have implemented both MS-DVCR and the original DVCR algorithm in the Java-based extensible simulator YAES [14]. We are considering the following experimental scenario. A square area of $L \times L$ meters is covered with sensor nodes deployed randomly and uniformly, with an average density of 0.004 nodes / m^2. The transmission range of each node is 30 meters. The nodes establish a system of a 4-dimensional directed virtual coordinates with the 4 anchor points being chosen as the nodes at the extreme edges of the deployment area. The sensor nodes make periodic observations which they report to the single mobile sink which is moving inside the deployment area.

For the experiments in this paper we consider a mobility model where the goal of the mobile sink is to move in such a way as to avoid the premature exhaustion of the energy at the nodes in its neighborhood. The objective of such a movement pattern is to equalize the load in terms of energy consumption the sink node inflicts in its neighborhood. This energy load is proportional to the number of received messages. Thus, the mobile sink will perform one movement step after a fixed number of received messages (this value, in our experiments, had been set to 20). In a movement step the sink will always move from one of the static nodes to one in the neighborhood. Overall, the mobile sink implements a random waypoint movement, with each movement broken down to a number of individual steps. The sink will, however, not perform an extra pause at the waypoints (as this would negate the objective of equalizing the stay in various neighborhoods).

In this scenario we have investigated two algorithms:

- **UA-DVCR** - is a simple extension of the DVCR routing algorithm for a scenario with a mobile sink. In this algorithm, the mobile sink will Update All the nodes in the network whenever it moves by broadcasting its coordinates to all the nodes in the network.
- **MS-DVCR** - our proposed algorithm, as described in Section III.

Table I lists the parameters of the experimental setup.

B. Energy consumption and average route length function of the size of the sensor network

The first series of experiments compare the energy consumption and the average route length function of the width of the sensor area. The sensor area width was varied between 400 and 1000 meters. With the constant deployment density, this means that the number of sensor nodes was varied between 640 and 4000 respectively. The energy consumption was calculated using the Rappaport communication model [15] and summed over all the nodes in the network. The average route length was calculated as the average of the number of hops traversed by the messages received by the sink. The experiments had been averaged over 10 different runs with different random seeds for the deployment of the nodes and the movement of the sink (however, the MS-DVCR and UA-DVCR protocols had been both run on all the specific random scenarios).

For these experiments, MS-DVCR had been run with a local area of radius $r = 4$.

Figure 2 shows the average path length in terms of the number of hops for the two protocols. The error bars show the 95% confidence interval. Our initial expectation was that the UA-DVCR will perform better here by yielding shorter paths. As in MS-DVCR the sink does not update the whole network about its new position when moving inside the local area, the overall routing tables might not be optimal. It turns out, however, that the differences between the path lengths in these protocols are barely visible and are essentially masked by the randomness of the generated network. As expected, for both protocols, the average path length increases with the size of the area.

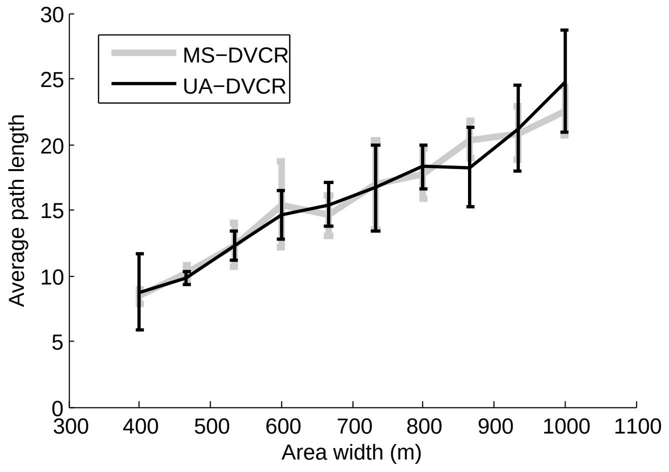


Fig. 2. Average path length for UA-DVCR and MS-DVCR function of area size.

Figure 3 shows the overall energy consumption of the two protocols. The error bars show the 95% confidence interval. As expected, the energy consumption of UA-DVCR is significantly higher than the one of MS-DVCR due to the overhead of the update messages broadcasted to the whole network. Furthermore, the energy consumption overhead in UA-DVCR increases faster than linearly with the number of nodes in the network (in our case, quadratically with the area width). In contrast, for MS-DVCR the majority of updates are broadcasted only in the fixed size local area. Studying the confidence intervals of the two graphs we can make the interesting observation that the energy consumption of MS-DVCR is highly predictable as it does not depend on the movement pattern of the sink. In contrast, the movement pattern of the sink, specifically, the number of times it leaves the local area significantly influences the energy consumption, which varies in a wider range. However, the values always stay significantly below the UA-DVCR values.

Overall, the conclusion of this set of experiments is that MS-DVCR closely matches the path length of UA-DVCR and achieves this with a significantly smaller energy consumption.

C. Routability function of the size of the sensor network

The frequent changes in the routing tables in a sensor network with a mobile sink will inevitably lead to messages which are not delivered. This typically happen when a message crosses the wave of the broadcasts delivering the new location of the sink. In this series of experiments we investigate whether this is a significant problem for UA-DVCR and MS-DVCR, as well as how the MS-DVCR compares with UA-DVCR on this metric.

Figure 4 shows the number of successfully delivered messages function of the size of the network. The results are averaged over 10 runs with different random seeds for the

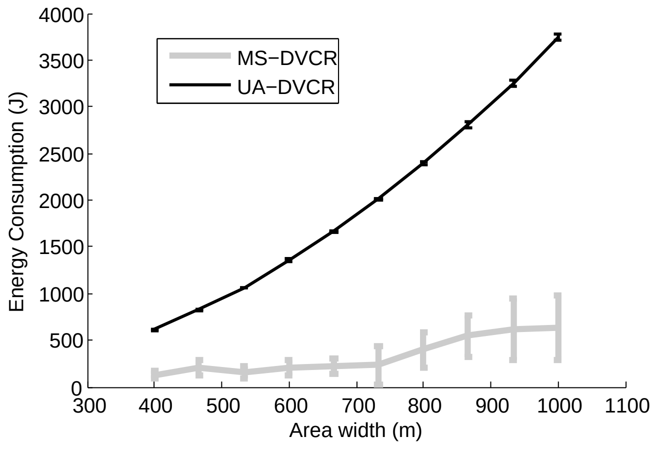


Fig. 3. Overall energy consumption for UA-DVCR and MS-DVCR function of area size.

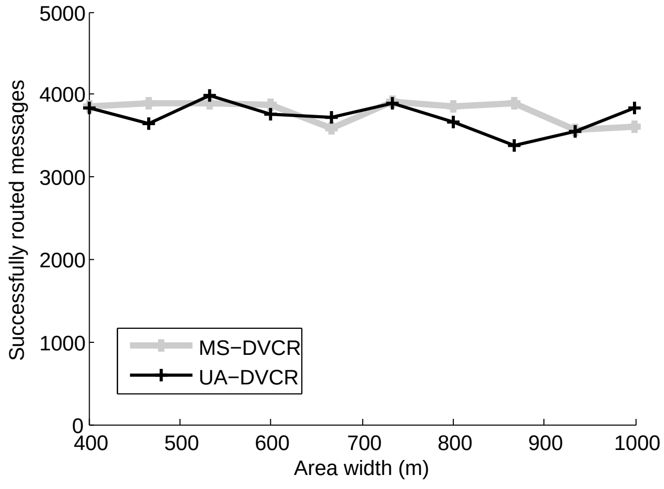


Fig. 4. Number of successfully routed messages for UA-DVCR and MS-DVCR function of area size. Total number of transmitted messages were 4000.

deployment of the nodes and the movement of the sink. To put the numbers in perspective, we need to remember that in total 4000 messages had been generated in each of these experiments.

We find that despite the sink's relatively fast movement (moving after every 20 received messages) the routability values are above 90% (and, in fact above 95% for most values). There is no clear correlation between the size of the area and routability. The experiments also fail to delineate a clear winner between the two algorithms. MS-DVCR appears to have a very slight benefit, possibly due to the fact that it performs global routing table updates more rarely.

D. Energy consumption function of the size of the local area

The size of the local area of the MS-DVCR protocol as expressed in its radius r influences the energy consumption of the network through two counteracting effects. First, the larger the r, the more expensive the local updates will be (for a value of r larger than the diameters of the network in hops, the MS-DVCR protocol is equivalent to the UA-DVCR protocol). On the other hand, the smaller the r the more often the sink will leave the local area, thus triggering an external update. The optimal size of the local area depends both on the relative cost of the local and external updates as well as the movement pattern of the sink.

In this series of experiments, we studied the energy consumption of the MS-DVCR protocol while fixing the other parameters to a network with a side length of 600m. As expected, areas which are too small yield a high energy consumption (due to frequent external updates), while large areas increase the energy consumption due to the higher cost of local updates. The optimal size of the local area (for these parameters) had been found to be $r = 5$.

One of the conclusions we can draw from this experiment, however, is that the MS-DVCR algorithm is highly sensitive to the choice of the parameter r - even a difference of 1 hop in the local area size can increase the energy consumption by 30% or more. As the optimal size depends on the movement pattern of the sink, the best practice for a real world deployment would be to calibrate the value of r by performing simulations of the real world network with realistic movement patterns of the sink.

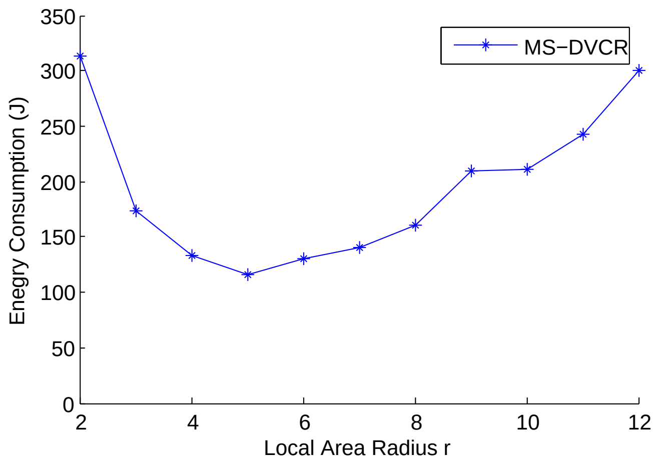


Fig. 5. Energy consumption based on the radius of destination area

V. Conclusions

In this paper, we have introduced MS-DVCR, a routing protocol specialized to routing towards a mobile sink in a network with directional virtual coordinates. The central idea of the protocol is to limit the updates sent out by the moving sink to a *local area*. An experimental study had shown that with respect to average path length the MS-DVCR protocol closely matches the performance of a simple extension of the DVCR protocol for mobile sinks, but with a significantly lower energy consumption. A separate set of experiments had shown that the performance of the MS-DVCR protocol is highly dependent on the appropriate choice of the local area size. The proposed protocol is significant also in that it demonstrates an approach for dealing with mobile nodes without the need for physical distance measurements.

References

[1] B. Karp and H.-T. Kung, “GPSR: Greedy perimeter stateless routing for wireless networks,” in *Proc. of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom-00)*, pp. 243–254, 2000.

[2] A. Caruso, S. Chessa, S. De, and A. Urpi, “GPS free coordinate assignment and routing in wireless sensor networks,” in *Proc. of 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM-05)*, vol. 1, pp. 150–160, 2005.

[3] Q. Cao and T. Abdelzaher, “Scalable logical coordinates framework for routing in wireless sensor networks,” *ACM Transactions on Sensor Networks (TOSN)*, vol. 2, no. 4, pp. 557–593, 2006.

[4] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica, “Geographic routing without location information,” in *Proc. of the 9th Annual International Conference on Mobile Computing and Networking, IEEE MobiCom ’03*, pp. 96–108, 2003.

[5] D. C. Dhanapala and A. P. Jayasumana, “Topology preserving maps from virtual coordinates for wireless sensor networks,” in *Proc. of the 35th IEEE Conference on Local Computer Networks (LCN-10)*, pp. 136–143, 2010.

[6] D. C. Dhanapala and A. P. Jayasumana, “Directional virtual coordinate systems for wireless sensor networks,” in *Proc. of IEEE International Conference on Communications (ICC-11)*, pp. 1–6, 2011.

[7] L. Bölöni, D. Turgut, S. Basagni, and C. Petrioli, “Scheduling data transmissions of underwater sensor nodes for maximizing value of information,” in *Proc. of the IEEE Global Communications Conference (Globecom-2013)*, 2013.

[8] R. C. Shah, S. Roy, S. Jain, and W. Brunette, “Data mules: Modeling and analysis of a three-tier architecture for sparse sensor networks,” *Ad Hoc Networks*, vol. 1, no. 2, pp. 215–233, 2003.

[9] L. Tong, Q. Zhao, and S. Adireddy, “Sensor networks with mobile agents,” in *Proc. of IEEE Military Communications Conference (MILCOM-03)*, vol. 1, pp. 688–693, 2003.

[10] L. Bölöni and D. Turgut, “Should I send now or send later? A decision-theoretic approach to transmission scheduling in sensor networks with mobile sinks,” *Wireless Communications and Mobile Computing Journal*, vol. 8, no. 3, pp. 385–403, 2008.

[11] J. Luo and J.-P. Hubaux, “Joint mobility and routing for lifetime elongation in wireless sensor networks,” in *Proc. of 24th Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE INFOCOM ’05*, vol. 3, pp. 1735–1746, 2005.

[12] G. Wang, T. Wang, W. Jia, M. Guo, H.-H. Chen, and M. Guizani, “Local update-based routing protocol in wireless sensor networks with mobile sinks,” in *Proc. of IEEE International Conference on Communications (ICC-07)*, pp. 3094–3099, 2007.

[13] D. C. Dhanapala and A. P. Jayasumana, “Clueless nodes to network-cognizant smart nodes: Achieving network awareness in wireless sensor networks,” in *Proc. of Consumer Communications and Networking Conference, IEEE CCNC ’12*, pp. 174–179, 2012.

[14] L. Bölöni and D. Turgut, “YAES - a modular simulator for mobile networks,” in *Proc. of the 8th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM’05)*, pp. 169–173, October 2005.

[15] T. S. Rappaport, *Wireless communications: principles and practice*. Publishing House of Electronics Industry, 2004.