Circular update directional virtual coordinate routing protocol in sensor networks

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, skhan, turgut, lboloni}@eecs.ucf.edu
† Department of Electrical and Computer Engineering,
Colorado State University, Fort Collins, CO
Email: anura.jayasumana@colostate.edu

***Abstract*—In a wireless sensor network, virtual coordinates provide most of the advantages of geographic routing strategies without actually relying on the location information of the nodes. Using a mobile sink provides advantages such as distributing energy consumption throughout the network. However, nodes need to be updated about the new virtual coordinate of the mobile sink as it moves. In this paper, we propose Circular Update-Directional Virtual Coordinate Routing (CU-DVCR), an algorithm specialized in routing towards a mobile sink in virtual coordinates. Through a set of experimental studies we show that CU-DVCR consumes less energy compared to alternative algorithms while providing comparable performance.**

I. Introduction

In wireless sensor networks, geographic routing protocols allow sending of messages without knowledge of the network topology. However, obtaining geographic information requires localization techniques such as GPS, which is costly or unfeasible in some applications. The virtual coordinate system (VCS) helps to build operational networks without the need for geographical localization. VCS relies upon the hop-by-hop distance information from a few anchor nodes [1], [2], [3], [4]. Directional virtual coordinate systems [5] adds directionality to VCS and provides a routing strategy, comparable to the geographical routing algorithms.

A simple way to gather data from nodes is to use a single sink node with fixed position. However, in recent years many applications utilize mobile sinks with help of technologies such as autonomous underwater vehicle (AUV) [6], unmanned aerial vehicle (UAV) [7], [8], and electric vehicles (e.g., Segway) [9], [10]. Although in many applications the sink movement is controlled and predictable [11], [12], [13], we need a routing strategy for the packets to reach the mobile sink. There exist strategies for routing towards mobile sinks in geographical domain [14], [15]. A solution which had been investigated both in the geographical domain [16] and virtual coordinate system [17] is to update the nodes in a local area around the sink instead of broadcasting the location of the sink to the entire network.

In this paper, we argue that even broadcasting the location information of the sink in a local area is not always the most energy efficient strategy. The Circular Update Directional Virtual Coordinate Routing (CU-DVCR) protocol described in this paper is basically sending the location information of the sink only to the nodes on the *boundary* of the local area. This allows us to update the packets entering the local area and to gradually notify all the nodes in the local area.

The remainder of this paper is organized as follows. Section II explains the geometrical construction and calculations underlining a virtual coordinate system and discuss directional virtual coordinates and the DVCR [5] routing algorithm. The CU-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 and DVCR

In virtual coordinates nodes are defined by their hop-count distance to a set of nodes called *anchor nodes*. In a sensor network with N nodes and M anchors, $h_{N_iA_j}$ shows the minimum hop distance between node N_i and anchor A_j. Thus, the virtual coordinate of node N_i is $[h_{N_iA_1}, \ldots, h_{N_iA_M}]$. As $h_{N_iA_j}$ is the same for all the nodes within a certain distance from A_j in all directions, it does not provide a sense of directionality. Thus, we need to use more than one anchor distance for a node to provide directionality information.

Table I shows a simple one dimensional network with two anchors in virtual coordinates. $h_{N_iA_1}+h_{N_iA_2}$ does not provide directionality for the nodes between the anchors since all the values are the same. On the other hand, $h_{N_iA_1}-h_{N_iA_2}$ is the same for all the nodes between the anchors. However, Equation 1 provides a directional coordinate for a one dimensional network with two anchors.

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

where $\frac{1}{2h_{A_2A_1}}$ is used for normalization.

For any two arbitrarily chosen anchors from all the anchors in the network, say A_j and A_k, let $\vec{f}(h_{N_iA_j}, h_{N_iA_k})$ be the ordinate:

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

TABLE I
A ONE DIMENSIONAL NETWORK CONSISTING OF CONSECUTIVELY CONNECTED NODES: N_1, N_2, A_1, N_3, N_4, A_2, N_5, N_6

	N_1	N_2	A_1	N_3	N_4	A_2	N_5	N_6
$h_{N_iA_1}$	2	1	0	1	2	3	4	5
$h_{N_iA_2}$	5	4	3	2	1	0	1	2
$h_{N_iA_1} - h_{N_iA_2}$	-3	-3	-3	-1	1	3	3	3
$h_{N_iA_1} + h_{N_iA_2}$	7	5	3	3	3	3	5	7
$f(h_{N_iA_1}, h_{N_iA_2})$	-3.5	-2.5	-1.5	-0.5	0.5	1.5	2.5	3.5

where $\vec{u}_{A_jA_k}$ is called the virtual direction and is the unit vector in direction of A_jA_k. Hence, the virtual distance between two nodes N_p and N_q in this direction would be defined as:

$$F_{A_jA_k}(N_p, N_q) = f(h_{N_pA_j}, h_{N_pA_k}) - f(h_{N_qA_j}, h_{N_qA_k}) \quad (3)$$

Let us now define the distance metric used in DVCR. Suppose that the transformed ordinates of the source node x and the sink node y are given as $N_x \equiv [n_{x1} \cdots n_{xj} \cdots n_{xP}]$ and $N_y \equiv [n_{y1} \cdots n_{yj} \cdots n_{yP}]$. Here P is the cardinality of the transformed coordinates and can be selected from $\binom{M}{2}$ combinations given M randomly selected anchors. Using the L^2 distance between the source node s and the destination d, we find the distance as

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

We can use this metric to perform greedy forwarding. When a node needs to transmit a message to the destination, 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,

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

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:

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

Similarly using reference anchor nodes A_3 and A_4 we get,

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

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 CU-DVCR ROUTING PROTOCOL

The nodes in a sensor network need to know the location of the sink to forward the messages to it. In sensor networks with a static sink, routing is possible once nodes know the location of the sink. In the case of a mobile sink, however, nodes need to be continuously updated about the new location of the sink. By considering the nodes inside a limited *local area* around the sink, we can use one of these solutions when the sink moves:

- **Update All-DVCR (UA-DVCR)** [17] - Update all the nodes in the network
- **Mobile Sink-DVCR (MS-DVCR)** [17] - Update the nodes in a local area around the sink
- **Circular Update-DVCR (CU-DVCR)** - Update the nodes on the boundary of a local area around the sink

The main idea behind MS-DVCR is to limit the radius of broadcasting to the nodes inside a local area while the sink is inside the area. CU-DVCR takes a step further by limiting the broadcasting to the nodes on the *boundary* of the local area. This idea comes from the fact that only nodes on the boundary of the local area are involved in correcting the assumption of incoming messages to the local area about the current location of the sink.

Let us assume that at the beginning of the scenario the sink is at the location $N_s = [n_{s1} \dots n_{sP}]$. The local area of the sink R is defined as the set of nodes with the L^2 distance to the sink smaller than r:

$$R = \{N_i \mid D_{N_iN_s} \leq r\} \quad (8)$$

The boundary of the local area B is defined as the set of nodes with an L^2 distance to the initial location of the sink between r and $r - c$ where c is the width of the boundary area:

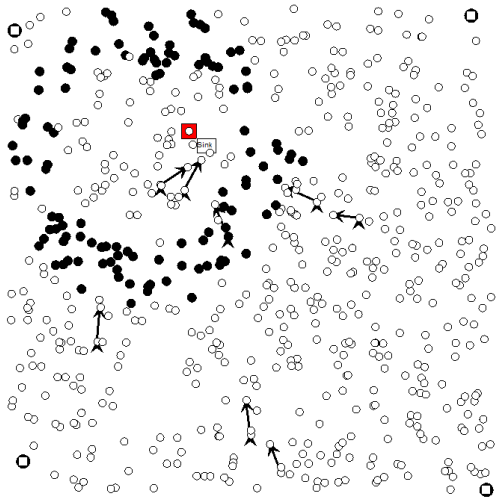
$$B = \{N_i \mid r - c \leq D_{N_iN_s} \leq r\} \quad (9)$$

By this definition of local area R and boundary area B, the sink can make two different types of moves:

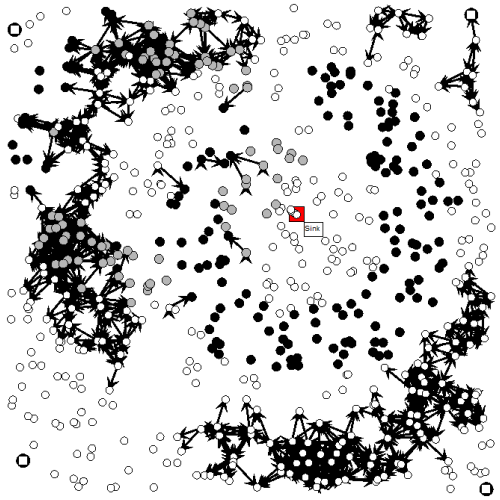
- **Local move**: the sink stays inside the local area. In this case, the sink will send a message to one of the nodes in B and that message will be broadcast to the rest of the nodes in B.
- **External move**: the sink leaves the current local area. Hence, it must (a) create a new local area, (b) specify the new boundary nodes and (c) broadcast the coordinates of the new local area center to the entire network. See Figure 1.

Now we can define three types of messages used in CU-DVCR considering different movements of the sink.

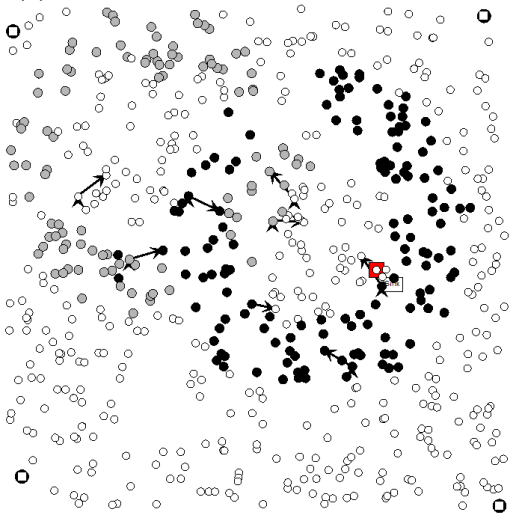
- **LOCAL messages** are sent by the sink and forwarded to the boundary of the local area. Once they reach one of the nodes of B, they will be broadcast to all the nodes of B. These messages carry the current location of the sink.
- **EXTERNAL messages** are broadcast to the entire network. They carry the location of the center of the newly



(a) Regular operation



(b) Network notification after external move



(c) New and previous boundary areas

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

formed local area, and they are sent when the sink performs an external move.

- **SENSING messages** are sent by the sensor nodes and carry sensed data to the sink. When these messages reach a node of B, they obtain the current location of the sink, and update all the encountered nodes on their way towards the sink.

Algorithm 1 describes the event-driven behavior of the sink in CU-DVCR. When the sink moves, it will calculate its new coordinates based on the coordinates of its neighbors. Then, it checks whether the new location is inside the current local area. If yes, it sends a LOCAL message to one of its random neighbors to be forwarded to the boundary area. When the sink leaves the local area, it creates a new local area. The current location of the sink will be chosen as the center of the new local area. Then, it broadcasts an EXTERNAL message containing this location information. The sink also receives SENSING messages and treats them according to the behavior specified in the application.

Algorithm 1 Sink behavior in CU-DVCR

```
when move do
    new-location := current location of sink
    if (D_L2(new-location, local-area-center)) < r  then
        send(msg(LOCAL, new-location), random-neighbor)
    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 nodes in CU-DVCR. When a node receives a LOCAL message containing the new location of the sink, it sets one of its neighbors as the next hop to the sink based on DVCR algorithm. Then it checks if it is inside the local area or not by calculating the distance between current node and the local area center. If it is inside the local area, it checks whether it is on the boundary area of the local area or not. If it is inside the boundary area, it will broadcast the message, otherwise, it will forward the message to the furthest node from the sink. All distance calculations are done in virtual coordinates using Equation 4 presented in Section II.

When a node receives an EXTERNAL message containing the new local area center, it updates itself, chooses the next hop to the sink, and broadcasts the message. A sensor node forwards a SENSING message to the next hop while receiving an update about the new sink location. It also creates a message after sensing data and forwards it to the next hop.

Let us now discuss the expected energy consumption of the UA-DVCR, MS-DVCR and CU-DVCR algorithms. We can divide energy consumption into three parts:

- E_{IL}: for updating the nodes inside the local area
- E_{OL}: for updating the nodes outside the local area

Algorithm 2 Node behavior in CU-DVCR

```
when receives(message(LOCAL, new-sink-location)) do
  nexthop := closest neighbor to new-sink-location
  if (D_{L^2}(local-area-center, nodelocation) < r) then
    if (D_{L^2}(local-area-center, nodelocation) > r-c) then
      broadcast(msg(LOCAL, new-sink-location))
    else
      send(msg(LOCAL, new-sink-location), farthest-neighbor-
      from-sink)
    end if
  end if
end when
when receives(message(EXTERNAL, new-local-area-center)) do
  local-area-center := new-local-area-center
  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
```

- E_S: for forwarding the sensed data to the sink

Since both MS-DVCR and CU-DVCR update the nodes outside the local area only when the sink does an external move, they are expected to have the value E_{OL} smaller compared to UA-DVCR. CU-DVCR is expected to have a lower value for E_{IL} than MS-DVCR and UA-DVCR since it is not using broadcasting to update all the nodes inside the local area. However, the events created at the nodes inside the local area may traverse a lengthier path due to lack of awareness about the exact location of the sink. These packets will be first forwarded to the previous location of the sink, and then forwarded to the new location. We expect the effect of increase in path length to be small on the total energy consumption due to two reasons: (a) the nodes which are not updated when the sink does a local move constitute a small portion of all the nodes in the network (b) these nodes will gradually receive the updated location of the sink as soon as a packet that has passed the boundary of the local area passes through them.

IV. Experimental study

A. Experimental setup

In order to demonstrate different aspects of CU-DVCR, we compare it with MS-DVCR and UA-DVCR which are the state-of-art algorithms with the same goal in virtual coordinates. The algorithms are implemented in the Java-based extensible simulator YAES [18]. Table II lists the parameters of the simulated scenarios. The parameters are set based on the constant values column of the table unless explicitly mentioned. In each experiment we vary one of the parameters in the range shown in the varying range column to see its effect on the desired metrics.

TABLE II
Experimental parameters

Parameter	constant values	varying range
General		
Sensor network area L (m)	800	400-1000
Node deployment	random uniform	-
Density (nodes / m^2)	0.005	0.002-0.006
Number of sensor nodes	5000	800-5000
Transmission range (m)	30	-
Sink movement	random waypoint [19]	-
Sink speed (m/s)	4	1-7
Experiment length	4000 messages	-
Number of runs	20	-
Protocols		
Coordinates	directed virtual	-
Anchors	4, extreme corners	-
local area radius r	10	5-17
boundary area width c	4	-

The sensor network in the experiments is a square area of $L \times L$ meters with sensor nodes randomly and uniformly distributed throughout the network. The virtual coordinates used 4 dimensions, with the sensor nodes at each corner of the square area serving as the anchor points. A single sink is moving in the area using a random waypoint movement model [19] at a constant speed. We used the energy consumption model of wireless communication as described by Rappaport [20]. For all the experiments described here, the results had been averaged over 20 different runs with different random seeds for the deployment of the nodes and the movement of the sink.

B. Energy consumption function of the size of the sensor network

In this section, we set up the experiments to compare the energy consumption function of the width of the sensor network. The sensor network width varies between 400 and 1000 meters with a constant deployment density to create networks with the number of sensor nodes between 800 and 5000. The sink was moving at 4m/s in a local area with the radius of 15 hops.

Figure 2 shows the overall energy consumption of the compared protocols. The energy consumption of UA-DVCR is higher than the other algorithms especially in larger networks due to higher number of messages required to update the entire network. CU-DVCR is more energy efficient than MS-DVCR in all network sizes. This is because the number of messages required to update the entire local area in MS-DVCR is larger than the number of messages to only update the boundary of local area in CU-DVCR. As explained in the previous section, the lenghthier path traversed by the messages in CU-DVCR has less impact on total energy consumption compared to E_{IL}, the energy used to update the nodes inside the local area.

C. Routability function of network density

The number of successfully delivered messages to the sink is an important factor in comparing routing algorithms. Figure 3 shows the number of successfully delivered messages as a function of network density. All the parameters are set according to Table II except sensor node deployment density

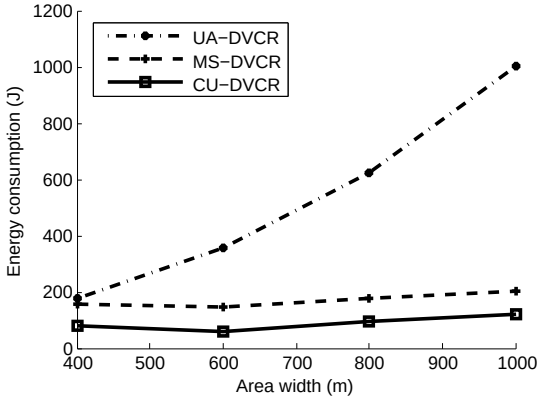


Fig. 2. Overall energy consumption for UA-DVCR, MS-DVCR and CU-DVCR function of area size.

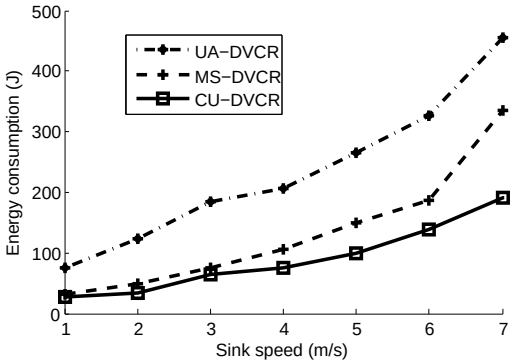


Fig. 4. Energy consumption for UA-DVCR, MS-DVCR and CU-DVCR function of sink speed.

which varies between 0.002 and 0.006 nodes/m^2. As expected, the higher node density causes an increase in the number of successfully delivered messages. In densities higher than 0.0055 nodes/m^2, 100% of the messages were successfully delivered to the sink. The three algorithms behaved essentially identical in this experiment.

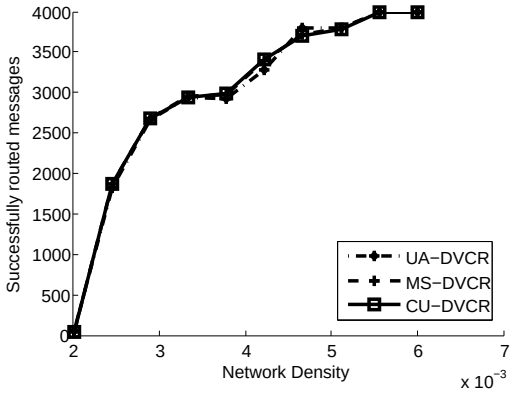


Fig. 3. Number of successfully routed messages for UA-DVCR, MS-DVCR and CU-DVCR function of network density.

D. Energy consumption function of the sink speed

In this section, we investigate the effect of varying the speed of sink movement on energy consumption. The parameters are set as explained in Table II. The sink speed varies from $1m/s$ to $7m/s$. Figure 4 shows the results. The faster the sink moves, the more updates are occurring in all the protocols, thus the energy consumption always increases with the sink speed. However, the area covered by the updates is largest for UA-DVCR (the whole network), smaller for MS-DVCR (the local area) and smallest for CU-DVCR (a circular area around the local area). While for a slowly moving sink this difference is minimal, as the sink moves faster, the updates represent a larger fraction of the total energy consumption of the network. For $7m/s$ the energy consumption of CU-DVCR is about 35% of UA-DVCR and about 60% of MS-DVCR.

E. Energy consumption and average path length function of the size of local area

In this section, we investigate the effect of varying the size of local area r on energy consumption and average path length. The average path length was calculated as the average number of hops that successfully routed messages traversed to reach the sink. The parameters are set as explained in Table II. In this experiment, the radius of local area radius r varies between 5 to 17 hops.

Figures 5 and 6 show the results. Choosing a small value for the radius of local area causes an increase in energy consumption in both MS-DVCR and CU-DVCR. This is because the sink leaves a smaller local area more frequently than a larger one. On the other hand, making the local area larger requires more number of packets to update it in MS-DVCR. Therefore, we can see that the energy consumption of MS-DVCR increases for the local areas larger than 9 hops. In CU-DVCR, the energy consumption decreases when the local area becomes larger. This is because the sink leaves the local area less often and also there is no broadcasting to the entire local area as in huMS-DVCR. However, this energy conservation is achieved at the expense of longer average path length towards the sink for the SENSING messages (see Figure 6).

One of the limitations of MS-DVCR is that we cannot conserve energy more than a certain limit by changing the radius of local area. However, in CU-DVCR this is possible at the expense of longer average path length. Therefore, in applications where the life-time of the network is critical, using a large r in CU-DVCR can be helpful. On the other hand, in time-sensitive applications where extra energy consumption can be tolerated, we can use a small r in CU-DVCR or even use MS-DVC with a large r to achieve shorter average path length.

V. Conclusions

In this paper, we introduced CU-DVCR, a routing algorithm towards a mobile sink in virtual coordinates. The main idea is to update the nodes on the boundary of a local area around the sink as the sink moves. We have compared this algorithm

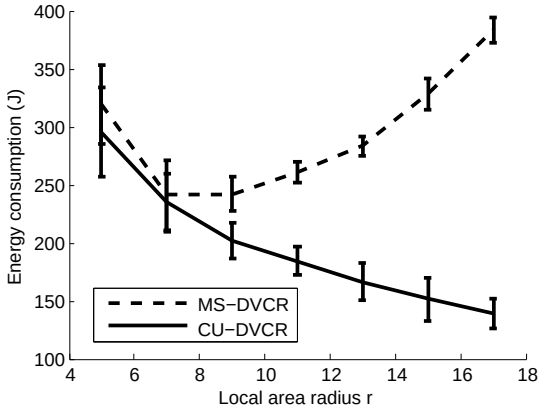


Fig. 5. Energy consumption in MS-DVCR and CU-DVCR function of the radius of local area r

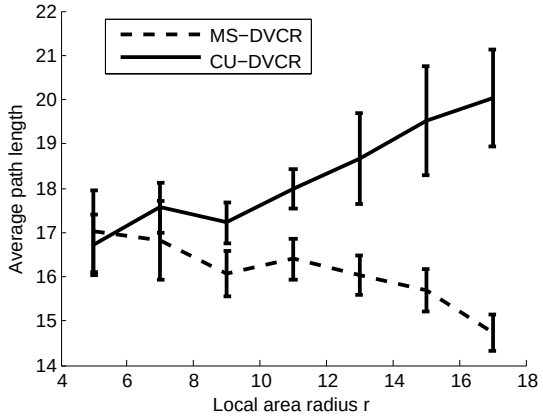


Fig. 6. Average path length in MS-DVCR and CU-DVCR function of the radius of local area r

with the state-of-art algorithms designed for the same purpose. Experimental studies show that CU-DVCR is more efficient in terms of energy consumption compared to other algorithms while maintaining comparable performance in the number of successfully routed messages. However, this result is achieved at the expense of somewhat longer average path length for the messages containing sensed data. We also showed that when the sensor node deployment density is enough, CU-DVCR guarantees 100% successful delivery of packets. The simulation results also indicate that the energy conservation in CU-DVCR is more pronounced for larger networks and faster sink movement.

REFERENCES

[1] 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)*, vol. 1, pp. 150–160, March 2005.

[2] 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.

[3] 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 (MobiCom)*, pp. 96–108, September 2003.

[4] M. Chen, X. Wang, V. Leung, and Y. Yuan, "Virtual coordinates based routing in wireless sensor networks," *Sensor Letters*, vol. 4, no. 3, pp. 325–330, 2006.

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

[6] J. Xu, G. Solmaz, R. Rahmatizadeh, D. Turgut, and L. Bölöni, "Animal monitoring with unmanned aerial vehicle-aided wireless sensor networks," in *Proc. of IEEE Conference on Local Computer Networks (LCN)*, October 2015.

[7] F. A. Khan, S. A. Khan, D. Turgut, and L. Bölöni, "Scheduling multiple mobile sinks in underwater sensor networks," in *Proc. of IEEE Conference on Local Computer Networks (LCN)*, October 2015.

[8] F. A. Khan, S. A. Khan, D. Turgut, and L. Bölöni, "Greedy path planning for maximizing value of information in underwater sensor networks," in *Proc. of IEEE Conf. on Local Computer Networks Workshops (LCN Workshops)*, pp. 610–615, 2014.

[9] G. Solmaz, K. Akkaya, and D. Turgut, "Communication-constrained p-center problem for event coverage in theme parks," in *Proc. of IEEE GLOBECOM*, pp. 486–491, December 2014.

[10] G. Solmaz and D. Turgut, "Optimizing event coverage in theme parks," *Wireless Networks (WINET) Journal*, vol. 20, pp. 1445–1459, August 2014.

[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 (INFOCOM)*, vol. 3, pp. 1735–1746, March 2005.

[12] G. Solmaz and D. Turgut, "Tracking evacuation of pedestrians during disasters," in *Proc. of IEEE GLOBECOM*, December 2015.

[13] G. Solmaz and D. Turgut, "Event coverage in theme parks using wireless sensor networks with mobile sinks," in *Proc. of IEEE ICC*, pp. 1522–1526, June 2013.

[14] J. Luo, J. Panchard, M. Piórkowski, M. Grossglauser, and J.-P. Hubaux, "Mobiroute: Routing towards a mobile sink for improving lifetime in sensor networks," in *Distributed Computing in Sensor Systems*, pp. 480–497, Springer, 2006.

[15] X. Li, J. Yang, A. Nayak, and I. Stojmenovic, "Localized geographic routing to a mobile sink with guaranteed delivery in sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 30, no. 9, pp. 1719–1729, 2012.

[16] 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)*, pp. 3094–3099, June 2007.

[17] R. Rahmatizadeh, S. Khan, A. Jayasumana, D. Turgut, and L. Bölöni, "Routing towards a mobile sink using virtual coordinates in a wireless sensor network," in *Proc. of IEEE ICC 2014*, pp. 12–17, June 2014.

[18] 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.

[19] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile computing*, pp. 153–181, Springer, 1996.

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