TRACK: A Novel Connected Dominating Set based Sink Mobility Model for WSNs

Avinash Srinivasan and Jie Wu
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431
Email: {asriniva@, jie@cse.}fau.edu

***Abstract*—The core functionality of a wireless sensor network (WSN) is to detect deviations in expected normal behavior and report it to the sink. In this paper, we propose TRACK — a novel sink mobility model exploiting the Connected Dominating Set (CDS) property of a network graph. TRACK, to the best of our knowledge, is the first contemporary sink mobility model to exploit the CDS property for WSN lifetime longevity and secure data aggregation. In TRACK, the CDS of the given network is computed and then the Minimum Spanning Tree (MST) of the CDS is constructed. Using the CDS-MST as the underlying framework, a Hamiltonian Circuit (HC) is constructed, along which the sink is mobilized to traverse the network. Since TRACK, by the very definition of CDS, passes through the transmission range of every node in the network, data can be relayed directly from the source node to the sink, eliminating the need for multi-hop routing. By virtue of this property, nodes in the WSN are discharged from their routing obligations and data aggregation becomes more secure. Additionally, we propose an extended version of TRACK called M-TRACK in this paper. The extended model trades higher fractions of sensor energy with the objective of minimizing the length of sink trajectory. This consequently minimizes the delay between consecutive sink visits, mitigating buffer-overflow of sensors. M-TRACK does necessitate multi-hop routing, but keeps it within a bounded number of hops. We confirm the efficiency and robustness of our models via simulation and analysis, and show that our model can extend the WSN lifetime up to seven times that which can be achieved using a static sink.**

Keywords: Connected dominating set (CDS), data aggregation, Hamiltonian circuit (HC), minimum spanning tree (MST), mobile sink, trajectory, wireless sensor networks (WSNs).

I. Introduction

Wireless sensor networks (WSNs) have become the epitome of pervasive technology, operating in environments typically not encountered by their counterparts. Sensor nodes have to be physically small to be pervasive, and have to be manufactured economically since they are deployed in large numbers and almost never reclaimed. Consequently, WSNs formed on-the-fly by ad-hoc networking of such small, inexpensive devices do not command the luxury of extensive computing and battery power. In literature, the usefulness of a WSN has typically been measured as the duration of time over which it delivers its intended services, which is referred to as the network lifetime. The lifetime of a WSN is typically measured as the duration from the initial deployment of the network until the failure of the first sensor. Therefore, it is essential to prolong the life of each and every sensor in the network [11] in order to prolong the lifetime of a WSN. While some researchers have proposed extending the lifetime of a WSN by using a mobile element [8], several others have employed multiple mobile elements for the same [10]. There is another class of researchers who propose using multiple static sinks at optimal locations for extending the lifetime of WSNs [9].

Sensor nodes, with their limited transmission capabilities, have to seek the cooperation of other nodes to relay data to the sink by employing a secure multi-hop routing protocol. One obvious drawback of multi-hop routing is that it leads to dissimilar rates of battery power depletion of nodes across the network. Specifically, nodes in the vicinity of the sink are subject to rapid battery depletion since they have to forward the data from hundreds of thousands of nodes in the network. On the otherhand, nodes away from the sink are seldom required to forward data and hence last longer. To address the aforementioned problem of uneven power depletion, sink mobility has often been employed as an effective solution. Since, with mobility, the sink moves to new locations, either being time-driven or event-driven, different nodes come into the sink's vicinity, which forces them to assume the role of a relay node and forward data. Some benefits of employing mobile sinks are as follows:

- Reducing the distance over which the data is relayed reduces the energy expended by multiple sensors enroute.
- It enhances the security and integrity of data by limiting the number of intermediate sensors. With few or no intermediate nodes, there is little room for attacks like blackhole, wormhole, packet-dropping, modification, man-in-the-middle, etc.
- Sink mobility mitigates collision and contention in the network, thereby enhancing system throughput. It also mitigates the *holes* [12] and *hot-spot* [5] problems in WSNs.
- With mobile sinks, sparse and disconnected regions in a network are covered efficiently. Even network partition problems can be resolved to a large extent.

The mobility model, TRACK, that we propose in this paper is based on the Connected Dominating Set (CDS) property of

This work was supported in part by NSF grants CCR 0329741, CNS 0422762, CNS 0434533, CNS 0531410, and CNS 0626240. Corresponding author's e-mail: asriniva@fau.edu

a network graph. The CDS is developed using the localized *Rule-k* algorithm proposed by Dai and Wu in [7]. Once the CDS of the given network graph is obtained, the Minimum Spanning Tree (MST) of the CDS is constructed using the Euclidian length of edges as the weighing criteria. For clarity, the Euclidian length of an edge $e(i, j)$ is the Euclidian distance between nodes i and j. After the MST is constructed, a Hamiltonian Circuit (HC) is developed for the sink to traverse the network [1]. However, sensors suffer from buffer overflow if the sink does not visit them in a timely manner. Hence, delay optimization is critical to network longevity. Therefore, we extend TRACK and propose MultiLevel-TRACK (M-TRACK) that trades higher fractions of sensor energy with the objective of shortening sink trajectory length to minimize the delay between consecutive sink visits.

Most existing protocols, reviewed in Section VI, either choose random mobility and let the sink traverse the network along a random trajectory, or opt for controlled mobility by defining the trajectory in terms of regular shapes such as a circle, a square, or a straight line. In this paper we advocate controlled mobility and propose a CDS-based trajectory. Intuitively, the trajectory along a CDS defines an optimal path since each node in the network is either in the CDS or within the range of a node in the CDS. Consequently, each node can communicate directly with the sink while it is moving along the CDS. With this as our base model, we propose an extended model, M-TRACK, that further optimizes the length of sink trajectory thereby minimizing sink visit delays. In summary, our contributions in this paper are as follows:

- TRACK is the first sink mobility model to exploit the CDS property of a network graph.
- CDS has been long employed for routing and coverage, but TRACK is the first to consider it for network longevity and secure data aggregation.
- We also propose an extended version of TRACK called M-TRACK based on MultiLevel-CDS to minimize buffer-overflow.
- We present algorithms for the construction of TRACK and M-TRACK and discuss them in detail.
- We validate the performance of our models through simulation and analysis.

The rest of the paper is organized as follows. In Section II, we give a brief overview on CDS and MST. In Section III, we present the overview of TRACK and its extended version M-TRACK. We also present formal algorithms for these models. In Section IV, we analyze TRACK, highlighting its strengths and salient features. We discuss the simulation results in detail in Section V. In Section VI, we review the related work and finally conclude the paper in Section VII.

II. Overview of CDS and MST

For the sake of completeness, we give a brief overview on CDS, MST, and HC in this section. We also briefly discuss the localized *Rule-k* algorithm proposed by Dai and Wu in [7]. Consider an undirected graph $G = (V, E)$, with V being the set of vertices and E being the set of edges. In G, a node u dominates another node v if and only if $u = v$ or u and v are adjacent. For example, in Fig. 1 (a), the set $\{1, 4, 13, 14\}$ is DS. Furthermore, let the CDS of G be a set of vertices V_{CDS} such that $V_{CDS} \subset V$. Now, we have the following:

Definition 1: A connected dominating set of a graph $G = (V, E)$ is a set of vertices $V_{CDS} \subset V$ such that for every vertex $v \in V - V_{CDS}$, there is at least one vertex $u \in V_{CDS}$ that dominates v, and V_{CDS} is connected.

For illustration, the set $\{1, 2, 4, 7, 8, 9, 10, 11, 13, 14\}$ forms the CDS of the network shown in Fig. 1 (a). Once the CDS is obtained, Dai and Wu's *Rule-k* algorithm is applied to reduce the size of the CDS. Using *Rule-k*, a node u can be unmarked from the CDS if u is completely covered by a subset of its neighbors N' and the following conditions are satisfied:

1) Subgraph induced by N' is connected.
2) Every neighbor of u is adjacent to at least one node in N'.
3) All nodes in N' have a higher priority than u.

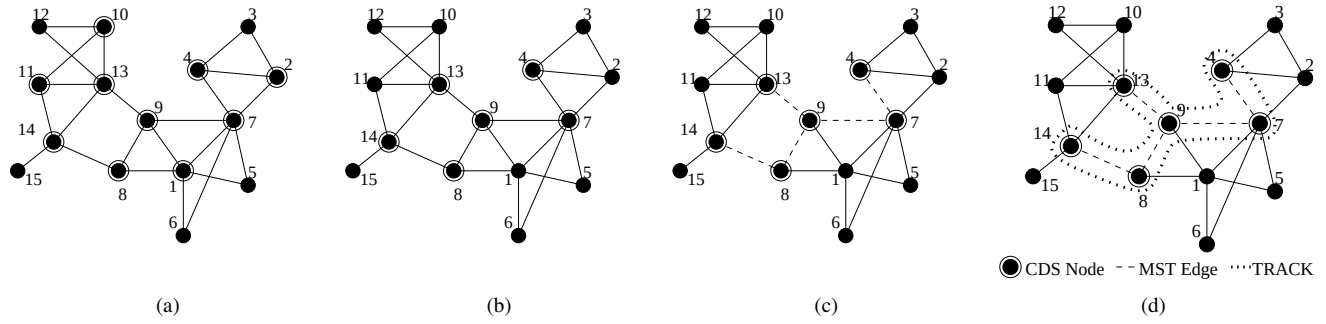
In our example, when *Rule-k* is applied to the CDS in Fig. 1 (a), CDS nodes 1, 2, 10, and 11 are pruned, resulting in a smaller CDS (40% smaller) $\{4, 7, 8, 9, 13, 14\}$, as shown in Fig. 1 (b). Note that the CDS can be constructed using the vertex ID, vertex degree, remaining battery power, or a combination of any of these as the priority value when inducing nodes into the CDS. The CDS constructed in Fig. 1 (a) is based on vertex ID priority.

Definition 2: Given a connected, undirected graph $G = (V, E)$, a spanning tree of G is a subgraph G' which is a tree that connects all the vertices in V together. An MST is a spanning tree G'' with a weight less than or equal to the weight of every other spanning tree G' of G.

Definition 3: Given a connected, undirected graph $G = (V, E)$ and its CDS-MST, a Hamiltonian Circuit (HC) of the CDS-MST is a path that visits each edge exactly once and returns to the starting vertex.

We have slightly adapted the definition of HC in Definition 3 to our model requirements.

For illustration, consider the network with the CDS as shown in Fig. 1 (b). In our model, the MST of the CDS is developed using distributed MST algorithm proposed by Gallager et al in [2], hereafter referred to as GHS-MST. The advantage of this distributed approach is that an MST can be formed even if not every node has a complete topology map. Fragments of an MST are created in a distributed fashion. Initially, G consists of $|V|$ fragments. Each fragment selects its minimum weight outgoing link, and via control messaging, each fragment arranges to merge with a neighboring fragment over its minimum weight outgoing link. This algorithm is shown to produce an MST in $O(|V| \times log|V|)$ time provided the edge weights are unique. For the sake of discussion, we assign unique random weights in terms of Euclidian length for each edge in E_{mst}. Let $d(9, 13) = 3$, $d(8, 9) = 4$, $d(4, 7) = 5$, $d(8, 14) = 6$, $d(7, 9) = 7$, and $d(13, 14) = 8$. Initially MST=$\{\emptyset\}$. Individual fragments are merged resulting in the MST as shown in Fig. 1 (c). Finally, an HC, as shown in Fig. 1 (d), is developed by traversing each link of the MST bidirectionally. Due to space limitations, we will not discuss the GHS-MST algorithm in detail, but interested readers may refer to [2].



(a) (b) (c) (d)

Fig. 1. (a) CDS before applying *Rule-k* (b) CDS after applying *Rule-k* (c) CDS-MST (d) TRACK: Trajectory of the sink along the HC of the CDS-MST.

Algorithm 1 INITIALIZE

for each sensor node $u \in V$ **do**
  for each sensor node $w \in V : w \neq u$ **do**
    if $u \in RNG_w$, $w \in RNG_u$, and $l(w,u) \notin E$ **then**
      $E \leftarrow E \bigcup l(u,w)$;

Algorithm 2 CDS

$INITIALIZE$;
Execute Localized *Rule-k* Algorithm on G to get V_{CDS};
$E_{CDS} \leftarrow \emptyset$
for each i, $j \in V_{CDS}$ **do**
  $E_{CDS} \leftarrow E_{CDS} \bigcup l(i,j)$

III. Mobility Models

In this section we present our mobility models, TRACK and M-TRACK, that we develop by exploiting the CDS property of a network graph. We also present formal algorithms for these models and delineate their working in detail.

A. TRACK

In TRACK, initially the CDS algorithm proposed in [7] is executed on the given network graph to compute the CDS. After the CDS is computed, GHS-MST is executed on the CDS. The resulting MST is used as the baseline in developing an HC as the trajectory for the sink to traverse the network. For illustration, consider Fig. 1 (a), which shows the CDS of a WSN with 15 sensors and Fig. 1 (b) shows the CDS of the same after applying *Rule-k*. In this example, the CDS consists of nodes $\{4, 7, 8, 9, 13, 14\}$. From Fig. 1 (c), we see that the MST of the given CDS consists of the links $\{l(4,7), l(7,9), l(9,13), l(8,9), l(8,14)\}$.

We define the vertex set V of the graph as $V = V_{reg} \bigcup V_{CDS}$, where V_{reg} is the set of non-CDS nodes and V_{CDS} is the set of CDS nodes, and $V_{reg} \bigcap V_{CDS} = \emptyset$. We define the edge set E of the graph as $E = E_{reg} \bigcup E_{CDS}$, where E_{reg} is the set of edges between non-CDS nodes or a CDS and a non-CDS node, and E_{CDS} is the set of edges between CDS nodes. For instance, in Fig. 1 (b), $(1,6) \in E_{reg}$ since 1 and $6 \in V_{reg}$; $(1,9) \in E_{reg}$ since $1 \in V_{reg}$ and $9 \in V_{CDS}$; and $(7,9) \in E_{CDS}$ since 7 and $9 \in V_{CDS}$. Edges in the set E_{CDS} can be further divided into E_{mst} and E_{nonMst}. In our previous example, as shown in Fig. 1 (c), $(13,14) \in E_{nonMst}$ and all other edges in E_{CDS} belong to E_{mst}.

From Definitions 1 and 2, the MST of a CDS is a tree that connects all vertices in V_{CDS} with the edges in E_{CDS}. The MST of the graph in our example is shown in Fig. 1 (c) as dashed edges. The path for the sink that is developed along E_{mst}, which we refer to as TRACK, is depicted in Fig. 1 (d).

Once the MST is obtained, traversing it to obtain the sink trajectory is accomplished as follows. There are three lists *Half-Open* (HO) and *Full-Open* (FO) keep track whether or not each link has been traversed bidirectionally, and *Traj* to keep track of the edge traversal order. The MST is traversed by starting at the highest ID node and adding the shortest out-going edge. When a leaf node is reached, the incoming edge is traversed back immediately. The FO list is always scanned first to find the shortest out-going edge from the current vertex. When the FO list is empty, the shortest out-going edge from the current vertex is chosen from the HO list. This has been formally presented in Algorithm 4. When all edges have been moved to the $Traj$ list and the current vertex is the starting vertex, the HC has been successfully constructed.

For example, consider Fig. 1 (c). In the discussions that follow, a link $l(i,j) = e(i,j) \bigcup e(j,i)$. For instance, in Fig. 1 (c), $l(7,9) = e(7,9) \bigcup e(9,7)$. Initially, $FO = \{l(4,7), l(7,9), l(9,13), l(9,8), l(8,14)\}$, $HO = \emptyset$, and $Traj = \emptyset$. Starting with 14, which is the highest ID node in V_{CDS}, $e(14,8)$ is added to $Traj = \{e(14,8)\}$, and $l(8,14)$ is moved to HO. At node 8, $e(8,9)$ is added to $Traj = \{e(14,8), e(8,9)\}$ and $l(8,9)$ is moved to HO. At node 9, there are two edges to choose from, $e(9,13)$ and $e(9,7)$. However, $e(9,13)$, being the shorter edge, is included in $Traj = \{e(14,8), e(8,9), e(9,13)\}$, and $l(9,13)$ is moved to HO. At 13, since there are no more out-going edges, $e(13,9)$ is traversed and added to $Traj = \{e(14,8), e(8,9), e(9,13), e(13,9)\}$, and $l(9,13)$ is deleted from HO. At node 9, either $e(9,8)$ or $e(9,7)$ can be chosen. However, since $l(8,9)$ is in HO and $l(7,9)$ is in FO, $e(9,7)$ is chosen and added to $Traj = \{e(14,8), e(8,9), e(9,13), e(13,9), e(9,7)\}$. Continuing on this line, we have $Traj = \{e(14,8), e(8,9), e(9,13), e(13,9), e(9,7), e(7,4), e(4,7), e(7,9), e(9,8), e(8,14)\}$, making it a full cycle, i.e., a Hamiltonian Circuit, as shown in Fig. 1 (d).

Once the trajectory is constructed, the sink moves onto

Algorithm 3 MST

$E_{mst} \leftarrow \emptyset$;
Execute GHS-MST on E_{CDS};
for each $l(i,j) \in E_{CDS}$ selected by GHS-MST **do**
$E_{mst} \leftarrow E_{mst} \bigcup l(i,j)$;

Algorithm 4 TRACK

$HO \leftarrow \emptyset, FO \leftarrow E_{mst}$;
max-ID$\leftarrow maxID(V_{CDS})$;
Extract $min(E_{mst})$ with *max-ID* as one of the vertices;
Delete the corresponding link from FO and move it to HO;
Continue extracting the shortest edge corresponding to the current V_{CDS} from FO
if $FO = \emptyset$ **then**
Extract the shortest edge corresponding to the current V_{CDS} from HO;
Delete the corresponding link from HO;
Move the corresponding link to HO and delete it from FO;

the trajectory and orbits it constantly. One obvious benefit of TRACK over several existing mobility models is that it eliminates the need for the sink to determine its next hop by simply following the edge order in $Traj$. The sink now only needs to compute its speed while moving along TRACK. Another benefit of TRACK over several existing models is that it effectively overcomes the hot-spot problem. Since, by moving along the CDS, the sink is within the communication range of every node in the network, multi-hop routing is eliminated. This also enhances the security of the message since the message is directly relayed by the sensing node to the sink. In this paper, we assume that the sink moves with a constant speed and collects data while moving. Upon sensing the sink in its range, each sensor node attempts to acquire the medium and transmit its data to the sink. We realize that this can lead to collision and contention. To resolve this problem we assume that a MAC layer protocol with appropriate back-off function is used in conjunction with TRACK.

Though TRACK is scalable, as the network size grows, the trajectory length, P_{length}, increases quickly. Consequently, the interval, t_{intrvl}, between sink visits at any given location on TRACK increases causing sensor buffer-overflow resulting in possible loss of critical data. To eliminate this problem, we recommend the deployment of multiple sinks along TRACK. With this, the buffer-overflow problem can be mitigated, and in most cases completely eliminated. However, the key concern when employing multiple sinks is determining the optimum number of sinks needed to eliminate the buffer-overflow problem. This can be examined through simulation studies, which we leave for investigation in our future work.

B. M-TRACK

We realize that TRACK can be further optimized in terms of P_{length}. To optimize P_{length}, we propose M-TRACK, which is a simple extension of TRACK, wherein the CDS algorithm is executed multiple times. For the first round, the given network is used as the input. For subsequent executions of the CDS algorithm, the CDS obtained from the previous round is used as the input node set. The stopping condition for the recursive call to the CDS algorithm is reached when the current call to the CDS algorithm cannot reduce the CDS from the previous round any further. The final output obtained at the stopping condition is a k-CDS, where k is the number of executions of the algorithm. The recursive call to the CDS algorithm is a tunable parameter and need not be executed until the stopping condition is reached. Once the desired level of CDS - referred to as k-CDS - is obtained with a sufficiently small |CDS|, the GHS-MST algorithm is executed, following which the HC is developed along the edges of the MST. The resulting HC serves as the sink trajectory and is referred to as M-TRACK. Note that if the trajectory is developed along the k-CDS for any value of $k > 1$, the P_{length} will be much smaller than it would be in TRACK.

To illustrate the impact of k-CDS on P_{length} for different values of k, consider Fig. 2. The 1-CDS of the network is shown in Fig. 2 (a), which consists of six nodes: $\{4, 7, 8, 9, 13, 15\}$. In Fig. 2 (b), the 2-CDS is shown for the same graph. We see that the 2-CDS, which consists of only three nodes $\{7, 8, 9\}$, is 50% smaller than that of 1-CDS. Here, note that for the 2-CDS, the 1-CDS is used as the input unlike 1-CDS for which the entire network is used as the input. The MST of our example network is shown in Fig. 2 (c), and the M-TRACK itself is shown in Fig. 2 (d).

In this particular example, the stopping condition is reached at 3-CDS which is not shown here. 3-CDS, in this example, would merely consist of node 9, which eliminates the need for sink mobility. The intuitive idea behind executing the CDS algorithm recursively is as follows. The first execution builds a 1-CDS which can be reached by all nodes in the network in a single hop. For instance, in Fig. 2 (a), we see that all nodes in the network can reach at least one of the 1-CDS nodes $\{4, 7, 8, 9, 13, 15\}$ in a single hop. Similarly, executing the CDS algorithm the second time builds a 2-CDS, which can be reached by every node in the network in at most two hops. From Fig. 2 (b), we see that every node in the network can reach at least one of the 2-CDS nodes $\{7, 8, 9\}$ in at most two hops. An important observation here is that all 1-CDS nodes should either be in 2-CDS or adjacent to nodes in 2-CDS. On similar lines, it can be argued for higher levels of CDS, which we represent as k-CDS. So, k essentially sets the upper-bound on the number of hops needed for routing the date from any sensor in the network to the sink.

In M-TRACK, the algorithm is recursively executed until a new CDS is obtained after each execution that covers all of the previous levels CDS nodes. Here we are attempting to achieve a shortened P_{length} and consequently a minimized t_{intrvl} by recursively building a smaller CDS. This ensures that the sink visits each location on M-TRACK in a timely manner, thereby avoiding buffer-overflow. However, this model has a downside. With multiple levels of CDSs, we are reintroducing routing as a necessity, coercing nodes to expend their energy in relaying data. This also jeopardizes the integrity of the data being relayed. However, we consider it to be more practical to trade-off a slightly large fraction of sensor energy per transmission to mitigate the problem of data loss. We believe that preserving

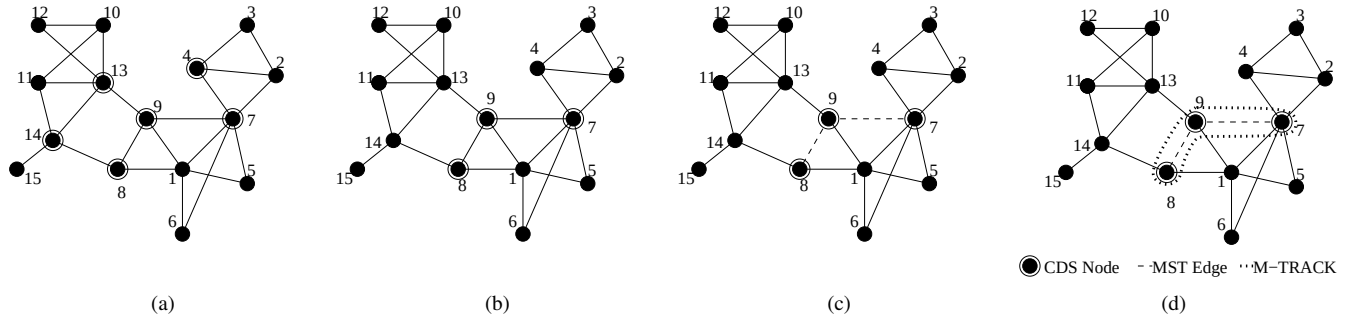


Fig. 2. (a) Level-1 CDS (b) Level-2 CDS (c) Spanning Tree of Level-2 CDS (d) M-TRACK: Trajectory of the sink along the level-2 CDS spanning tree.

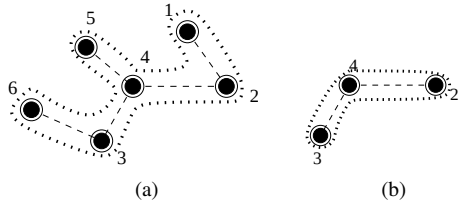


Fig. 3. (a) TRACK trajectory path-length; (b) MTRACK trajectory path-length.

critical sensed data is as important as extending the lifetime of the WSN itself.

IV. Analysis

For computing the path-length, P_{length}, of the sink trajectory, let us count the number of edges along the path. From Fig. 3 (a), we see that in TRACK, P_{length} is 10 for one complete cycle. In M-TRACK, this distance is reduced to 4 along 2-CDS as shown in Fig. 3 (b). As a direct observation, we see that the trajectory path-length in M-TRACK along 2-CDS is lowered by 60%.

The sink speed in our models can be computed as $Sink_{speed} = \frac{P_{length}}{t_{ovrflo}}$. Here, t_{ovrflo} is a tunable parameter and is adjusted according to the requirements of a particular network and the dynamics of the sensing field. From the above equation, the time interval between sink visits t_{intrvl} at any location on the trajectory can be computed as $t_{intrvl} = \frac{P_{length}}{Sink_{speed}}$. To overcome the buffer-overflow problem, the required condition is $t_{intrvl} \leq t_{ovrflo}$. The length of the trajectory, P_{length}, is very critical and has the most significant impact on the performance of the network. P_{length} in both of our models is computed as follows:

$$P_{length} = \sum_{\forall e(i,j) \in E_{mst}} len_{link}^{(i,j)} \tag{1}$$

where $len_{link}^{(i,j)}$ is the effective length that $l(i,j) \in E_{mst}$ adds to P_{length}. $len_{link}^{(i,j)}$ itself is computed as $len_{link}^{(i,j)} = 2 \times d(i,j)$, where $d(i,j)$ is the Euclidian distance between nodes i and j. P_{length} is computed as the sum of the lengths of all links in E_{mst}. For example, in Fig. 3 (a) we have $P_{length} = len_{link}^{(1,2)} + len_{link}^{(2,4)} + len_{link}^{(4,5)} + len_{link}^{(4,3)} + len_{link}^{(3,6)}$, where $len_{link}^{(1,2)} = d(1,2) + d(2,1)$, $len_{link}^{(2,4)} = d(2,4) + d(4,2)$, and so on. In summary, P_{length} can be computed as follows:

$$P_{length} = 2 \times \sum_{\forall e(i,j) \in E_{mst}} d(i,j) + \delta \tag{2}$$

Since the trajectory is slightly offset from the CDS nodes, in Equation 2, δ is used to compensate for the offset. The irregularly-shaped trajectory of the sink along the CDS is equivalent to a circular trajectory whose radius is given by:

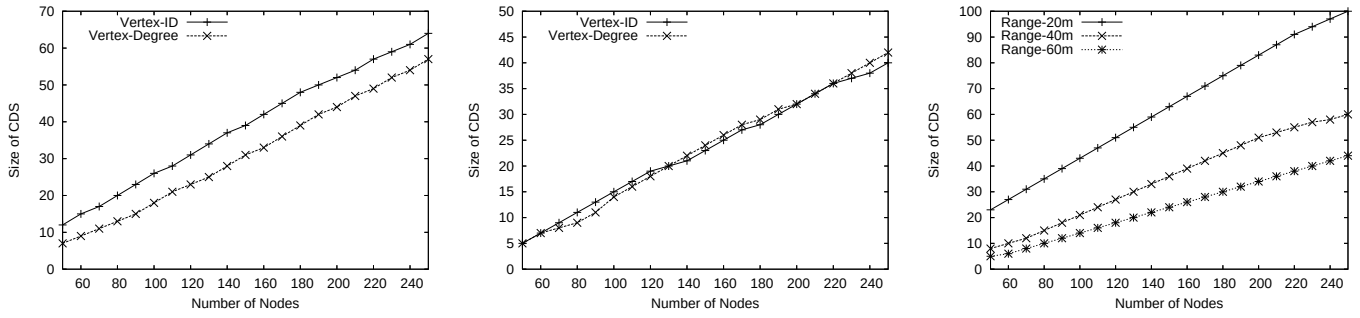
$$r = \frac{\sum_{\forall e(i,j) \in E_{mst}} d(i,j) + \delta}{\pi} \tag{3}$$

Equivalently, our irregular shaped trajectory can be expressed as a square whose side is given by $a = \frac{r}{2 \times \pi}$.

In [10], it is not possible to have a single mobile entity, while moving along a straight line trajectory, to gather data from all the sensors in the network in a single hop. However, in TRACK, a single sink can collect data from all the sensors in the network in one hop. In this aspect, TRACK is far more efficient compared to the model in [10]. Also, in [10], the sink is idle on its return journey to its starting point. But in TRACK and M-TRACK, the sink collects data continuously without idling. With M-TRACK, a shorter trajectory for the sink can be developed to mitigate the buffer-overflow problem.

However, M-TRACK necessitates multi-hop routing for the sensors to relay their data to the sink. However, the routing here is simple and easy to implement. Every node forwards its data to its dominating node. If a node has more than one dominating node, then it can randomly choose one of the dominating nodes to forward its packets to or use an underlying reputation monitoring system to monitor its dominating nodes and for routing data, use the most trustworthy dominating node. If a dominant node receiving the message is further dominated by a higher level CDS node, then it merely forwards its data as well as the data from the nodes' that it dominates to its dominant node. This process continues until the data reaches nodes that are within one-hop's distance from the sink's trajectory. The maximum number of hops in M-TRACK is a tunable parameter and hence gives leverage in optimizing the tradeoff between length of the trajectory and energy expenditure of sensors.

We realize that when nodes fail or move to a new region, or new nodes join, TRACK, along the current CDS, may not be the best path for the sink to traverse the network. This problem can be resolved by rebuilding the CDS, MST, and subsequently



(a) (b) (c)

Fig. 4. (a) Size of CDS with Fixed d=15 (b) Size of CDS with Fixed d=30 (c) Size of CDS with range varied from 20m to 60m.

the HC-based TRACK whenever such events occur. However, with frequent node failures, movements, and/or joins, it becomes impractical to rebuild the TRACK repeatedly. We will further investigate these scenarios in our future work. For now, we assume a static network with a fixed number of nodes in our simulations, and that no new nodes will be added to the network. With this, our only concern will be to address node failure. However, since we are measuring the network lifetime as the time from initial deployment of the network until the failure of the first node, node failure is a desired factor to evaluate the effectiveness of our model.

V. Simulation and Results

In this section, we first discuss the simulation environment followed by the results.

A. Environment

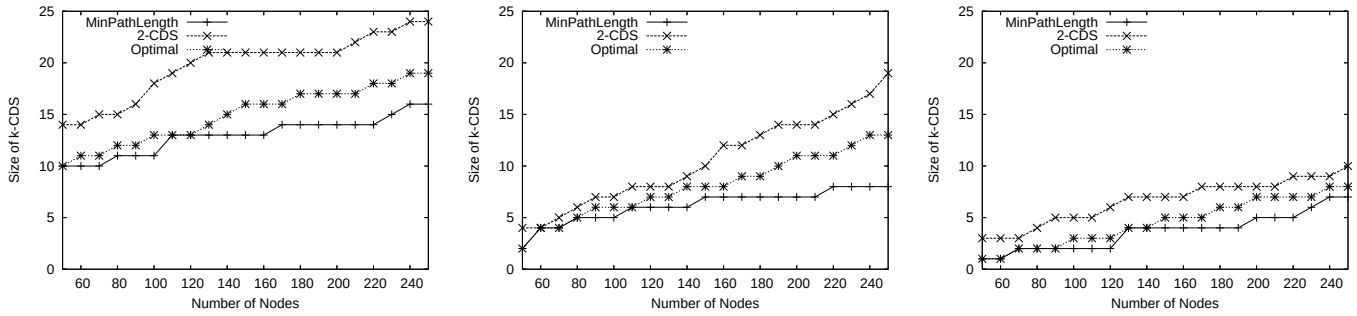
All simulations have been carried out on a custom built, stand-alone C++ simulator. In our simulations, a sensor field of area $100 \times 100\ m^2$ has been considered. The following parameters have been considered as tunable in our simulations: number of sensor nodes N_{sen}, number of sink nodes N_{sink}, and transmission range R_i of sensor node i expressed in meters. The network has been modeled as an undirected graph $G = (V, E)$, where V is the set of nodes and E is the set of edges between these nodes. An edge exists between two nodes u and v, if u lies in RNG_v and v lies in RNG_u. Each link $l(u, v) \in E$ is considered to be bidirectional. The results presented in this section have been averaged over 1000 iterations for statistical stability.

B. Results

We have studied the impact of different parameters on the size of the CDS. First, we fix the diameter of the network to $d = 15$ and accordingly vary the transmission range of the sensors as the density of the network changes. Varying the range of the sensors with changing density of the network to keep the diameter constant helps us to understand the impact of network size independent of density. The results for $d = 15$ are presented in Fig. 4 (a). Similarly we have studied the impact of diameter $d = 30$ on the CDS size and the results are in Fig. 4 (b). We see that both these graphs have two curves each: one for the CDS formed using vertex ID as the priority value and the other with vertex degree as the priority value. We see that with higher diameter, the two priority criteria perform equally well. Also, we can see that with increasing diameter, the size of the CDS shrinks. With $d = 15$ and vertex ID as the priority, the CDS is on average about 29% the size of the network and with vertex degree as the priority, it is about 25% the size of the network. For $d = 30$, it is about 14.5% with both vertex ID and vertex degree as priority. The size of the CDS, with $d = 15$ and vertex ID as priority, is on average about 6% smaller that that with vertex degree, and with $d = 30$, it is below 0.5%. In the rest of our simulations, unless otherwise specified, vertex ID is used as the priority value in constructing the CDS.

We have also studied the impact of density on CDS size by fixing the transmission range. We have simulated three different scenarios varying the range from 20m to 60m in increments of 20m; the results are presented in Fig. 4 (c). It is evident that as the transmission range increases, the size of the CDS shrinks. Nonetheless, the size of the CDS increases with increase in the network size but at much slower rate with higher transmission range. With range fixed at 20m, the size of the CDS, on average, is about 42% of the size of the network. Similarly, with range fixed at 40m and 60m, the CDS size is 22% and 15% the size of the network respectively. In Fig. 5, we have presented the size of k-CDS for three different scenarios: minimum path-length where k can be any positive integer (*scenario-1*) usually greater than 2, path-length with 2-CDS (*scenario-2*), and the optimal path-length where again k can take any value of any positive integer (*scenario-3*). The difference between *scenario-1* and the *scenario-3* is that in the former, the CDS algorithm is recursively executed until the stopping condition is reached while in the latter, the stopping condition need not be reached. Fig. 5 (a) shows the size of k-CDS for the three scenarios with a transmission range of 20m. We see that the size of K-CDS for *scenario-2* is considerably smaller when compared to the size of k-CDS for *scenario-3*.

Similarly, in Fig. 5 (b) and (c), we have presented the results for the three scenarios with transmission range fixed at 40m and 60m respectively. We can see from the results that the size of the CDS, irrespective of the scenario under consideration, increases with an increase in the network size but decreases with an increase in the transmission range. *scenario-3*, with



(a) Range=20m (b) Range=40m (c) Range=60m

Fig. 5. Comparison of CDS size: minimum path-length, 2-CDS, and Optimal.

transmission ranges 20m, 40m, and 60m has performed well below *scenario-2*, but it is *scenario-1* that performs best as far as keeping the CDS size the smallest is concerned.

In Fig. 6 (a), we have presented the results that depict the improvement in the overall network lifetime achieved in both TRACK and M-TRACK. In the results presented, the transmission range of sensors is kept constant at 40m and a network with a static sink located at the center of the network with a lifetime value of 1 is used as the benchmark for evaluating the performance of our models. We see that M-TRACK outperforms TRACK as long as the network is less dense. M-TRACK quickly improves the network lifetime to nearly six times that of the benchmark and eventually stabilizes around the same value as the density increases. The performance of M-TRACK ceases to improve with increasing network density.

On the otherhand, with increasing density, the lifetime improvement achieved by TRACK quickly outperforms M-TRACK and starts to stabilizes at a value of 7. This is due to the fact that, with increase in network size, nodes along the M-TRACK are forced to forward data from other nodes in the network. Consequently, a *hot-spot* zone develops around the trajectory and nodes start to die quickly. This is where the benefits of one-hop transmission of TRACK become significant. Without any routing obligations, nodes tend to survive longer and the *hot-spot* problem does not arise.

In Fig. 6 (b), we have plotted the results comparing the network lifetime improvement achieved in M-TRACK with a fixed transmission range of 40m for varying values of k. We see that the M-TRACK achieves up to six fold improvement in network lifetime with $k = 2$. With $k = 3$, the system achieves nearly four times higher lifetimes for smaller network sizes but beyond a certain network size, it starts to decay, eventually bringing down the effective lifetime of a network to just over two fold. With $k = 4$, though it achieves higher lifetimes for smaller network sizes compared to when $k = 2$ and 3, it constantly decays bringing down the lifetime on par with that of a static sink for larger networks.

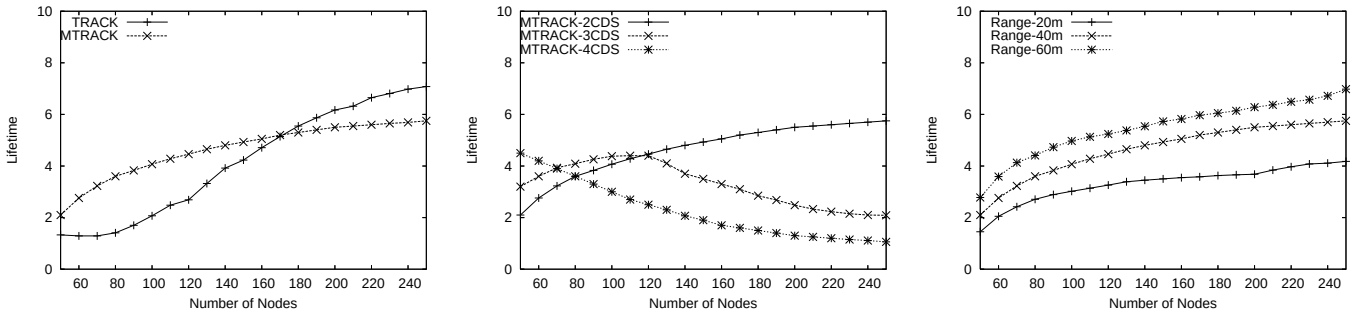
Finally, in Fig. 6 (c), we have plotted the results comparing the network lifetime improvement achieved in M-TRACK for varying transmission ranges with $k = 2$. We see that with the sensor transmission range fixed at 20m, the average lifetime improves approximately by two fold for a network of 50 nodes and to over four fold for a network of 250 nodes. Similarly, for a transmission range of 40m, the lifetime improves by two fold for a network of 50 nodes and to nearly six fold for a network of 250 nodes. For a transmission range of 60m, a maximum lifetime improvement of approximately three fold is achieved for a 50 node network and for a 250 node network the lifetime improvement is nearly seven fold. It is evident that with increasing transmission range, the lifetime of the network increases quickly in smaller networks when compared to larger networks. The lifetime improvement with 60m range is on average about 30% higher when compared to a 40m range and about 50% higher when compared to a 20m range. With 60m range, the lifetime improvement achieved in M-TRACK is on average about 6.5 times that of the benchmark model.

In summary, we have confirmed through simulations that both of our models, TRACK and M-TRACK, outperform a WSN with static sink(s). We have shown that our models can improve the network lifetime up to seven times that which is achieved in a network with a static sink. We have drawn meaningful results by comparing the size of the CDS and consequently the length of the sink trajectory for different transmission ranges. Similarly, we have drawn a meaningful relationship between network lifetime and network density.

VI. Related Work

Recently, mobile entities, mostly mobile sinks, have been used as data collection and processing elements to achieve network longevity in WSNs. Sink mobility in WNS can be classified into four groups as follows: random, predictable, controlled, and adaptive. Adaptive mobility is the least researched among the four types listed. In [3], Li and Rus propose a method for achieving guaranteed message delivery in minimal time. Their approach considers the modification in the mobile host trajectory and develop algorithms for minimizing the change in trajectory. In [4], the authors have considered random mobility of all the nodes for improving data capacity and have proved that two-hop routes are sufficient to achieve the maximum throughput capacity of the network.

In [6], a three-tier architecture for data collection in sparse sensor networks is proposed. This model exploits mobile entities called MULEs to pickup data from sensors in range and drop it off to the sink using a random walk mobility model. In [8], Kansal et al have proposed mobility control methods



(a) Range=40m (b) Range=40m (c)

Fig. 6. Lifetime improvement in (a) Track vs. M-TRACK (b) M-TRACK with varying k values (c) M-TRACK with varying ranges.

for improving network lifetime and data fidelity. The main deviation in their work comes from adding controlled mobile components to achieve predictable and larger gains. This idea is further extended in [10] where multiple mobile entities that move on a line are examined.

In [14], Chatzigiannakis, Kinalis, and Nikoletseas propose four models with characteristic mobility patterns for the sink along with different data collection strategies: (1) random-walk mobility with passive data collection, (2) partial random walk with limited multi-hop data propagation, (3) biased random walk mobility with passive data collection, and (4) deterministic walk with multi-hop data propagation. In [13], Luo and Hubaux investigate how to optimally move the sink along a circular trajectory. They consider joint mobility and routing algorithms, and show that a better routing strategy uses a combination of round routes and short paths. In [5], the authors make the first effort toward deterministic mobility of sink by using an integer-linear program to determine new locations for the sinks and a flow-based routing protocol to ensure energy efficient routing during each round.

In [9], Oyman and Ersoy focus on determining the optimal number of sinks needed and their location, given the minimum required operational time for the sensor network, based on clustering techniques. In [11], Wang et al propose a linear programming formulation for determining the movement of the sink and its sojourn time at different locations within the WSN such that maximum network lifetime is achieved. Their model prolongs the lifetime of a 256-node WSN up to almost five fold in comparison to a WSN with a static sink.

VII. Conclusion

In this paper, we have proposed TRACK, a novel sink mobility model based on the Connected Dominating Set (CDS) property of a network graph. We have also extended TRACK and proposed M-TRACK based on multi-level CDS. M-TRACK further enhances the performance of the network by optimizing the path-length of the sink trajectory by trading a slightly large fraction of sensor energy. We have presented algorithms for our models and confirmed the validity of both TRACK and M-TRACK through simulation and analysis. In our future work, we would like to apply the CDS property of a network graph to solve more problems in WSNs. We would also like to investigate the scan-based method for developing the Hamiltonian Circuit. However, this method comes at an additional cost since geographical locations are essential for the functioning of the scan-based method. Another investigation on our agenda for future work is employing the property of triangle-inequality for optimizing the sink trajectory path-length along the minimum spanning tree.

References

[1] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. Introduction to Algorithms, McGraw-Hill.

[2] R. G. Gallager, P. A. Humblet, and P. M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Transactions on Programming Languages and Systems*, vol. 5, no. 1, pp. 66-77, 1983.

[3] Q. Li and D. Rus. Sending messages to mobile users in disconnected ad-hoc wireless networks. *In the Proceedings of ACM MobiCom*, August 2000.

[4] M. Grossglauser and D. Tse. Mobility increases the capacity of ad hoc wireless networks. *In the Proceedings of IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 477486, 2002.

[5] S. R. Gandham, M. Dawande, R. Prakash, and S. Venkatesan. Energy Efficient Schemes for Wireless Sensor Networks with Multiple Mobile Base Stations. *In the Proceedings of IEEE GLOBECOM*, December 2003.

[6] R. C. Shah, S. Roy, S. Jain, and W. Brunette. Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks. *In the Proceedings of IEEE SNPA*, 2003.

[7] F. Dai and J.Wu. An Extended Localized Algorithm for Connected Dominating Set Formation in Ad Hoc Wireless Networks. *IEEE Transactions on Parallel and Distributed Systems*, October 2004.

[8] A. Kansal, A. Somasundara, D. Jea, M. Srivastava, and D. Estrin. Intelligent fluid infrastructure for embedded networks. *In the Proceedings of MobiSys*, 2004.

[9] E.I. Oyman and C. Ersoy. Multiple sink network design problem in large scale wireless sensor networks. *In the Proceedings of IEEE ICC*, June 2004.

[10] D. Jea, A. Somasundara, M. Srivastava. Multiple Controlled Mobile Elements (Data Mules) for Data Collection in Sensor Networks. *In the Proceedings of IEEE/ACM DCOSS*, June 2005.

[11] Z. M. Wang, S. Basagni, E. Melachrinoudis, C. Petrioli. Exploiting Sink Mobility for Maximizing Sensor Networks Lifetime. *In the Proceedings of HICSS*, 2005

[12] N. Ahmed, S. S. Kanhere, and S. Jha. The Holes Problem in Wireless Sensor Networks: A Survey. *In ACM SIGMOBILE Mobile Computing and Communications Review*, Vol. 1, No. 2, April 2005.

[13] J. Luo and J.-P. Hubaux. Joint Mobility and Routing for Lifetime Elongation in Wireless Sensor Networks. *In the Proceedings of IEEE INFOCOM*, 2005.

[14] I. Chatzigiannakis, A. Kinalis, and S. Nikoletseas. Sink Mobility Protocols for Data Collection in Wireless Sensor Networks. *In the Proceedings of ACM MobiWac*, October 2006.